Sculpture virtuelle par système de particules

Marc Helbling

Thèse

Pour obtenir le grade de

Docteur de l'Institut National des Sciences Appliquées de Rouen

Spécialité Mathématiques Appliquées

Présentée et soutenue à huis clos par

Marc Helbling

le 25 novembre 2010

Sculpture virtuelle par système de particules

Jury

Président	Jean-Pierre Pécuchet	Professeur, INSA de Rouen
Rapporteur	Bruno Lévy	Directeur de recherche, INRIA Nancy Grand-Est
Rapporteur	Olivier Gibaru	Professeur, Arts et Métiers ParisTech CER Lille
Examinateur	Carole Le Guyader	Maître de conférences, INSA de Rouen
Directeur de thèse	Christian Gout	Professeur, INSA de Rouen
Encadrant industriel	David Bonner	Ingénieur de Recherche, Dassault Systèmes

Thèse présentée au sein du Laboratoire de Mathématiques de l'INSA de Rouen (LMI)
et de l'École Doctorale SPMII

À la mémoire de Geneviève et de François.

À Fanny.
À ma famille.

« La liaison fortuite des atomes est l'origine de tout ce qui est. »
Démocrite

Remerciements

En premier lieu, je tiens à remercier Patrick Johnson, Pascal Sebah pour avoir rendue cette thèse CIFRE au sein de Dassault Systèmes possible. À leurs côtés s'ajoutent mon directeur de thèse, Christian Gout, et mon encadrant industriel, David Bonner, qui auront su m'encadrer en me laissant suffisamment de liberté pour que je puisse explorer mes idées.

Je suis également très reconnaissant aux membres de mon jury, en particulier Bruno Lévy et Olivier Gibaru, qui, malgré des agendas très chargés, m'ont fait l'honneur d'accepter de rapporter mes travaux de thèse.

Mes remerciements vont ensuite à l'ensemble de l'équipe Recherche de Dassault Systèmes pour sa bonne humeur générale, les échanges d'idées, et les soutiens d'anciens doctorants dans les moments de doute ! En particulier, ces travaux doivent beaucoup à Pierre-Édouard Cailliau pour les nombreuses conversations cosmiques et conseils en LaTeX, Emmanuel Tillaux pour son expertise *ès* tables de hachage, Cédric Soubrié pour la sensibilisation à l'ergonomie et aux besoins utilisateurs et aussi, pour l'aide à l'implémentation, aux petites mains de Denis Ok, Steren Giannini, Antoine Leblanc et Roland Brochard.

Cette thèse n'aurait certainement pas pu être finie sans le café de la Brûlerie Caron et sans la créativité de ces anglais qui pratiquent si bien la musique et qui ont su me redonner l'énergie quand elle semblait épuisée.

Pour finir, un grand merci à mes amis et à ma famille. L'émotion finale valait bien les sacrifices faits et la sueur écoulée. Et un énorme merci à ma souris d'avoir tout fait pour que je ne me casse pas les dents...

Table des matières

Table des figures

Introduction

La 3D émerge comme un nouveau média. Ce constat s'observe, entre autres, par l'essor des mondes virtuels, qu'ils soient réalistes (*e.g. Second Life*) ou fantaisistes (*e.g. World of Warcraft*), mais également par la possibilité, de plus en plus fréquente, de visualiser un objet sous n'importe quel angle, avant un achat électronique par exemple. L'utilisation de la 3D est ainsi en voie de démocratisation auprès du grand public. L'apparition de bases de modèles 3D, à l'instar de 3DVIA [Dasa] ou Google 3D Warehouse [Gooa], renforce ce constat.

Les médias traditionnels que sont la photographie et la vidéo ont connu un regain d'intérêt avec la popularisation d'appareils numériques et avec le développement d'outils de retouche numérique puis de sites internet communautaires d'échanges de contenu. L'adoption de la 3D par le grand public passe ainsi par la création d'outils performants de génération de contenu tridimensionnel et abordables par le peu, voire le non initié. Le matériel informatique étant de plus en plus abordable, certains logiciels, comme 3DVIA Shape [Dasb] (fig. 1) ou Google Sketchup [Goob], ont cherché à rendre la création de modèles 3D accessible au grand public.

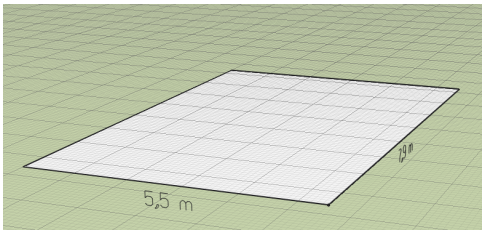
FIG. 1: Exemple de réalisation obtenue avec 3DVIA Shape.

Cependant, des tests utilisateurs, réalisés au sein de Dassault Systèmes avec des utilisateurs novices et sur des stations de travail standard, montrent la difficulté des personnes inexpérimentées à prendre conscience du fait que ces logiciels permettent de travailler des formes en trois dimensions. Cette difficulté peut notamment s'expliquer par trois aspects :

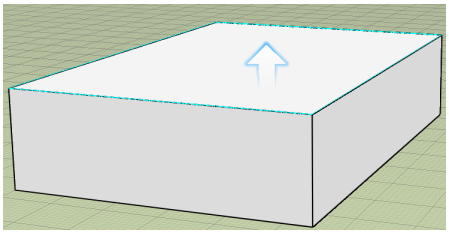
1. l'utilisation de périphériques 2D, écran et souris, pour contrôler et visualiser la 3D, *i.e.* des formes et scènes tridimensionnelles,

2. les modes d'interaction parfois peu naturels : la création d'un cube passe par un carré (fig. 2a) que l'on extrude pour obtenir un cube (voir fig. 2b),

3. difficulté d'appréhension de représentations géométriques dans l'espace.



(a) Création d'un rectangle



(b) Création d'un parallélépipède par extrusion.

FIG. 2: Processus d'obtention d'un cube avec 3DVIA Shape.

Même s'il existe aujourd'hui des solutions de visualisation en 3D, comme la solution holographique de Jones *et al.* [JMY+07], et que les écrans 3D grand public se démocratisent rapidement, l'écran 2D reste le périphérique standard pour le grand public et il faut donc s'en accommoder. Notons que ce point peut grandement influer sur la représentation spatiale mentale des utilisateurs. Les paramètres de caméra pour le rendu doivent donc imiter au mieux le mode de fonctionnement de l'oeil humain, notamment pour les perspectives. Le point 1 apparaît donc comme une contrainte matérielle imposée. Par conséquent, rendre la 3D plus accessible nécessite une réflexion sur les outils logiciels permettant d'améliorer les points 2 et 3.

L'objectif de nos travaux est la recherche d'un nouveau paradigme de manipulation *temps réel* de la 3D pour faciliter la création de formes. Par temps réel, on entend que l'utilisateur soit en mesure de visualiser sa forme à une fréquence au moins égale à la persistance rétinienne. L'intégration des remarques précédentes nous amène à considérer une modélisation :

- *intuitive*, reproduisant par exemple, des interactions du monde réel, ou d'outils virtuels globalement maîtrisés, à l'instar des outils de manipulation d'image ;

- *non limitative*, permettant à l'utilisateur de créer des objets formellement tridimensionnels sans brider son processus créatif à cause de considérations techniques.

Les outils actuels reposent généralement fortement sur la modélisation des surfaces qu'ils utilisent. Les manipulations possibles de formes sont alors peu intuitives ou limitées, voire les deux. Par exemple, les outils reposant sur des maillages polygonaux [BPK+07] ou des surfaces de subdivisions [ZS00] gèrent difficilement les changements de topologie. La modélisation volumique par *Constructive Solid Geometry* permet de créer des formes complexes mais nécessite des manipulations booléennes peu intuitives. Enfin, l'utilisation de surfaces polynomiales [Ris91] nécessite la manipulation indirecte et donc peu intuitive d'un polygone de contrôle ; par ailleurs, la nature polynomiale implique la définition par morceaux des surfaces complexes et nécessite la gestion, contraignante, des raccords de continuités entre morceaux

pour obtenir un résultat plaisant.

Finalement, il semble intéressant de chercher à abstraire la définition des formes 3D des contraintes liées à l'utilisation d'une modélisation particulière. Pour ce faire, il est intéressant de s'interroger sur les outils utilisés dans le monde réel pour créer des formes.

L'argile a été utilisée par les premiers hommes pour créer des oeuvres artistiques (fig. 3). De nos jours, l'argile a été remplacée par la pâte à modeler pour les enfants mais continue d'être utilisée, entre autres, par les *designers*. Les raisons de cette utilisation sont certainement liées à la présence du matériau, en quantités importantes, dans la plupart des régions du globe. Néanmoins, l'argile présente d'autres propriétés intéressantes. La matière peut devenir plus ou moins fluide et donc malléable en lui ajoutant de l'eau ; la contrepartie étant qu'en n'humidifiant pas le matériau, il sèche et finit par ne plus être manipulable. De plus, l'argile est très déformable et peut prendre n'importe quelle forme. Enfin, l'argile est un matériau non compressible.

FIG. 3: Figurine préhistorique en argile.

Aujourd'hui, il existe plusieurs variantes de matériaux de type pâte à modeler. On trouve notamment :

- des pâtes à modeler *lisses*, qui imitent fortement l'argile en enlevant la contrainte de fluidité (fig. 4a) ;
- des pâtes à modeler *par billes*, qui sont un amas de petites billes collées les unes aux autres (fig. 4b).

Si les deux types de matériaux conservent la faculté de représenter n'importe quelle forme, chacun présente des avantages et des inconvénients. Le matériau lisse est visuellement plus plaisant mais nécessite un recollement de matière à la jonction de plusieurs morceaux. A l'opposé, la pâte à modeler par billes est visuellement moins plaisante mais gère naturellement l'adjonction de morceaux. Idéalement, le matériau se comporterait comme la pâte à modeler par billes pour la manipulation et comme la pâte à modeler lisse pour la visualisation.

L'idée de simuler de la pâte à modeler virtuelle n'est pas récente. Dès 1988, Terzopoulos et Fleischer [TF88] parlent de « *computer plasticine* » :

(a) Pâte à modeler *lisse* (b) Pâte à modeler *par billes*

FIG. 4: Types de pâtes à modeler.

> *"We envision users, aided by stereoscopic and haptic input-output devices, carving computer 'plasticine' and applying simulated forces to it in order to create free-form shapes interactively."* [1]

Les auteurs proposent un modèle continu, basé sur des éléments finis, permettant de traiter les comportements tels que l'élasticité, la plasticité et la fracture mais incompatible avec une manipulation temps réel.

Galyean et Hughes [GH91] ont ensuite proposé une approche de sculpture de matériau virtuel par surface implicite. Les auteurs comparent leur méthode à celle d'outils de dessin en deux dimensions, avec un pinceau peignant de la matière dans des « pixels volumiques » . Les outils proposés permettent l'ajout, la suppression ou le lissage de matière.

Tonnesen [ST92] est le premier à introduire l'utilisation des systèmes de particules à des fins de sculpture [Ton98]. Il propose un modèle basé sur des particules orientées qu'il force à rester proches de la surface et sur lesquelles il peut reconstruire une peau.

Cani a travaillé sur plusieurs modèles, d'abord liés à la simulation de matériaux hautement déformables avec Desbrun [DC96] puis, plus spécifiquement, à la simulation de pâte à modeler virtuelle [CA06]. Le modèle purement implicite de Dewaele et Cani [DC04a, DC04b] repose sur une imitation de comportements physiques permettant la répartition de matière dans l'espace.

Enfin, Cristiano *et al.* [CFMU04] proposent une forme de synthèse entre le modèle de Tonnesen [Ton98] et le modèle de Dewaele et Cani [DC04a, DC04b] en utilisant un système volumique de particules ayant des interactions de type attraction-répulsion. Une surface implicite est extraite pour visualiser une forme continue.

Des conclusions ressortent de l'observation des matériaux réels et de cette brève revue de quelques modèles de matériaux virtuels :

1. intuitivement, une particule peut être vue comme un « *petit élément* » de matière. Une forme quelconque est un volume[2] qui peut donc être représenté par un système de particules.

   La topologie d'un système de particules pouvant être quelconque et variable dans le temps, il présente donc une solution non limitative pour la représentation d'une forme 3D. De plus, il est possible d'ajouter des comportements phy-

[1] « Nous imaginons des utilisateurs, utilisant du matériel stéréoscopique et haptique, manipulant de la plasticine virtuelle et lui appliquant des forces simultanées pour créer des formes libres interactivement. »

[2] Tout objet physique représente un volume, aussi petit soit-il.

siques à un système de particules de façon à obtenir des interactions proches de celles du monde réel. On peut ainsi chercher à reproduire un matériau de type pâte à modeler par billes ;

2. les fluides peuvent être considérés comme des corps hautement déformables puisque, par nature, leurs molécules ne sont pas fortement liées et donc leur forme n'est pas figée. Les fluides peuvent être la base d'un modèle de matériau déformable ;

3. la visualisation d'une surface représentant le système de particules permettra de donner un aspect continu au matériau à l'instar des pâtes à modeler lisses.

Le présent mémoire s'articulera de façon logique pour définir un outil de création de formes libres intuitif et non limitatif. De l'analyse précédente, nous avons fait le choix initial d'utiliser des systèmes de particules volumiques. Des travaux menés parallèlement aux nôtres [Bec09, Len09] montrent la puissance des modélisations particulaires pour la simulation réaliste de matériaux aux propriétés diverses et pouvant changer de caractéristiques.

Dans le premier chapitre, nous présenterons les outils retenus pour la modélisation des systèmes de particules volumiques en temps réel. Nous utiliserons ces outils pour définir des comportements de matériaux virtuels dans le second chapitre, et les problématiques de visualisation de ces systèmes seront abordées dans le troisième chapitre.

Afin de pouvoir combiner différentes approches et notamment réutiliser des modèles existants, nous verrons comment convertir des modèles surfaciques dans notre modélisation, dans le chapitre quatre.

Comme souligné par Gaylean et Hughes [GH91],

> *"it is often desirable to rough out the coarse shape of a sculpture first, and work on finer detail afterwards."* [3]

La création artistique passe généralement par deux phases : une première phase d'esquisse, dressant les contours généraux de la forme finale, puis une phase de finition. Nous étudierons donc enfin les outils à mettre en place pour l'esquisse et la finition de formes 3D libres.

Nous aborderons ainsi, dans le chapitre cinq, un ensemble d'outils permettant une manipulation locale et nécessaires à la sculpture. Ces outils de manipulation locale seront étendus par une approche de déformation globale dans le chapitre six.

Enfin, avant de conclure ces travaux, le chapitre sept présentera une réflexion sur l'ajout de détails sur nos modèles.

[3] « il est souvent souhaitable d'esquisser une sculpture grossière dans un premier temps, puis de travailler les détails ensuite. »

Première partie

Systèmes de particules

Systèmes de particules : Introduction

Nous souhaitons offrir la possibilité à l'utilisateur de concentrer son processus créatif sur la forme tridimensionnelle qu'il désire réaliser.

La détermination de la forme peut se faire par une modélisation dans laquelle l'utilisateur décrirait simplement l'agencement de la matière dans l'espace. L'utilisation d'un système volumique de particules, où chaque particule représente un « petit » élément de matière semble intéressante.

Nous allons donc chercher à définir les outils fondamentaux qui nous serons nécessaires pour la modélisation avec système de particules. Nous verrons ensuite comment employer ces outils afin d'ajouter des comportements physiques aux particules, permettant de simuler un matériau utilisable dans un contexte de sculpture.

Nous nous intéresserons enfin à la visualisation des matériaux continus représentés par des nuages volumiques de points.

Chapitre 1

Modélisation par systèmes de particules

Nous avons vu, en introduction, que l'utilisation d'un système de particules semblait un choix intéressant pour la modélisation de formes. Nous allons, dans ce chapitre, justifier ce choix d'approche, qualifiée de *lagrangienne*.

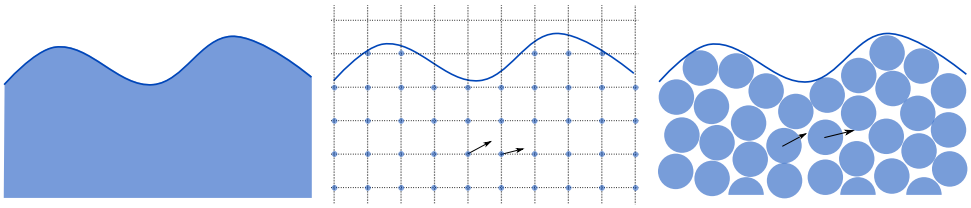
Notre objectif est d'obtenir des matériaux aux propriétés similaires à celles de la pâte à modeler. Nous définirons donc un ensemble d'outils permettant la simulation d'équations, dépendant de l'espace ou du temps, en approche lagrangienne. Enfin nous nous attacherons à présenter des algorithmes efficaces pour pouvoir effectuer ces simulations le plus efficacement possible, de façon à permettre des interactions utilisateur en temps réel.

1.1 Bref historique

Reeves [Ree83] fut le premier à introduire l'utilisation des systèmes de particules dans la communauté *computer graphics* pour la création d'effets spéciaux sur des objets « flous » comme du feu. Chaque particule, avec des propriétés qui lui sont propres, est simulée indépendamment des autres. Dans ce cas, un système de particules possède une source qui génère aléatoirement des particules avec une position initiale, une vitesse initiale, une durée de vie et d'autres attributs pour le rendu. À chaque pas de temps, la source crée de nouvelles particules et les particules existantes sont déplacées selon des lois *physiques*. La vie des particules est diminuée de la durée du pas de temps et les particules dont la durée de vie est nulle sont supprimées de la simulation.

Les approches à base de particules se prêtent bien à la modélisation d'objets non cohérents. Le mouvement de chaque particule peut être calculé indépendamment des autres puisque les particules n'interagissent pas entre elles, ce qui permet, en outre, une implémentation en parallèle.

Les systèmes de particules ont ensuite été étendus à des systèmes cohésifs pour la simulation de textiles. On parle également de systèmes masses-ressorts : les masses peuvent être vues comme des particules et les ressorts comme un lien entre les particules permettant de conserver la cohésion du système. Dans ce cas, on crée un réseau fixe reliant les particules entre elles, *i.e.* on fixe la topologie du système.



(a) Volume de fluide à simuler. (b) Approche eulérienne. (c) Approche lagrangienne.

FIG. 1.1: Approches pour la simulation de fluide.

Seule la géométrie, *i.e.* la position de chaque particule, est modifiée[1]. Ces modifications prennent évidemment en compte les liaisons entre les particules. Cette prise en compte demande une certaine robustesse lors de l'intégration des équations du mouvement comme démontré par Baraff et Witkin qui utilisent un schéma d'intégration implicite [BW98].

L'utilisation des systèmes de particules pour la simulation d'objets continus, comme les fluides, s'est alors démocratisée notamment grâce à la méthode *Smoothed Particles Hydrodynamics*, introduite dans les communautés *Computer Graphics* par Desbrun [Des97] et popularisée ensuite par Müller *et al.* [MCG03]. En effet, les systèmes de particules permettent de formuler les problèmes sous forme lagrangienne et la conceptualisation des particules en boules permet une gestion facile des collisions.

Aujourd'hui, les systèmes de particules sont utilisés dans de nombreux domaines, pour des simulations de matériaux continus allant de la simulation d'avalanche [Kro10] à la simulation de cheveux [BCN03] mais également pour des matériaux de nature discrète comme les poudres [BYM05].

Détaillons à présent le sens à donner à la dénomination d'approche *lagrangienne*, qui est le point de départ de notre modèle.

1.2 Approche lagrangienne

La revue des modèles de pâte à modeler, présentée en introduction, suggère l'utilisation de fluides pour simuler des matériaux hautement déformables. L'historique des systèmes de particules montre également qu'ils peuvent simuler des fluides.

La simulation de fluide nécessite de déterminer comment représenter le volume de fluide à simuler. Il existe deux familles d'approches (fig. 1.1a) :

1. approche *eulérienne* (fig. 1.1b) : les grandeurs physiques sont modélisées en tant que fonctions du temps et de l'espace. L'espace est discrétisé et les grandeurs physiques du fluide sont exprimées par rapport à un référentiel extérieur. Une grandeur physique s'écrit alors

$$A = \tilde{A}(t, \mathrm{x}),$$

où t représente le temps et x un point de l'espace ;

[1] la topologie peut également varier si on traite les cas de fracture par exemple.

2. approche *lagrangienne* (fig. 1.1c) : les grandeurs physiques sont modélisées en tant que fonctions du temps et de la matière. Le matériau est directement discrétisé en particules qui portent les grandeurs physiques. Une grandeur physique s'écrit
$$A = \hat{A}(t, P),$$
où P symbolise la particule matérielle d'évaluation.

Les écoulements fluides sont décrits en terme de *dérivée matérielle* $\frac{D\cdot}{Dt}$ définie par
$$\frac{D\,A}{Dt} \triangleq \left.\frac{dA}{dt}\right|_P.$$
Il s'agit de la dérivée totale à élément fixe du matériau *i.e.* une particule suivant l'écoulement fluide.

Commençons par écrire la différentielle en temps de la grandeur physique pour chaque approche. En approche eulérienne on a
$$\frac{dA}{dt} = \left.\frac{\partial \tilde{A}}{\partial t}\right|_{\mathrm{x}} + \left.\frac{\partial \tilde{A}}{\partial \mathrm{x}}\right|_t \frac{d\mathrm{x}}{dt},$$
et en approche lagrangienne
$$\frac{dA}{dt} = \left.\frac{\partial \hat{A}}{\partial t}\right|_P + \left.\frac{\partial \hat{A}}{\partial P}\right|_t \frac{dP}{dt}.$$

Considérons à présent un élément particulier du matériau, soit $P = P_O$. On a alors $\frac{dP}{dt} = 0$ et, on peut reconnaître l'expression de la vitesse $\mathrm{v}_{P_O} = \left.\frac{d\mathrm{x}}{dt}\right|_{P=P_O}$. Les dérivées matérielles s'écrivent finalement

- en approche eulérienne :
$$\frac{DA}{Dt} = \frac{\partial \tilde{A}}{\partial t} + \mathrm{v} \cdot \nabla \tilde{A}, \tag{1.1}$$
- en approche lagrangienne :
$$\frac{DA}{Dt} = \frac{\partial \hat{A}}{\partial t}. \tag{1.2}$$

L'expression de la dérivée matérielle est donc sensiblement plus simple en approche lagrangienne. De plus, l'approche eulérienne nécessite un découpage spatial qui induit une contrainte spatiale forte sur la simulation. L'approche lagrangienne, de par sa discrétisation du matériau lui même, ne présente pas cette limitation qui correspond mieux à nos objectifs. Tout le volume modélisé correspond au matériau lui même et représente ainsi un volume utile.

Nous utiliserons une approche lagrangienne et donc des particules de matière discrétisant un matériau. La dénomination *système de particules* sera, dans la suite, utilisée pour désigner la discrétisation en approche lagrangienne d'un objet en particules.

Une particule i sera généralement la simple donnée d'une position $\mathrm{r}_i(t)$, d'une masse m_i et d'une vitesse $\mathrm{v}_i(t)$. Nous considérerons des matériaux homogènes et des particules de même masse m afin de pouvoir ajouter le même niveau de détail

à tout endroit. De cette masse constante et de la densité volumique ρ_0 du matériau modélisé, on peut déduire un volume de matière représenté par une particule. Pour simplifier, on considère une boule dont le rayon est

$$r_p = \sqrt[3]{\frac{3m}{4\pi\rho_0}}.$$

Par défaut, on considère que les volumes associés aux particules dans un état stable sont tangents, *i.e.* l'espacement Δs des particules en état stable est

$$\Delta s = 2r_p.$$

Avant d'étudier les équations nécessaires à la simulation d'un écoulement fluide, nous avons besoin de certains outils permettant d'évaluer les grandeurs physiques portées par les particules, et d'un schéma d'intégration dans le temps pour effectivement simuler l'écoulement.

L'évaluation des grandeurs physiques peut être réalisée par la méthode lagrangienne d'approximation *Smoothed Particle Hydrodyanmics* que nous allons maintenant présenter.

1.3 Smoothed Particle Hydrodynamics

La méthode *Smoothed Particle Hydrodynamics*, que nous noterons *SPH* dans la suite de ce document, a été développée en parallèle par Lucy [Luc77] et Gingold et Monaghan [GM77] à la fin des années 70, initialement pour résoudre des problèmes d'hydrodynamique et d'astrophysique.

Le terme *Smoothed* résulte du fait que cette méthode fait intervenir une fonction noyau qui va lisser les grandeurs.

Le terme *Particle* découle de la modélisation lagrangienne utilisée et de la discrétisation du domaine d'intérêt en « particules » .

L'utilisation première de l'approche pour la résolution de problèmes de mécanique des fluides explique l'emploi historique du terme *Hydrodynamics*.

La plus grande généralité de la méthode a amené certains auteurs à étendre la méthode à d'autres domaines que la mécanique des fluides et à la renommer *Smoothed Particle Applied Mechanics*. D'un point de vue plus mathématique, la méthode peut être vue comme une méthode lagrangienne d'interpolation ; la terminologie *Smoothed Particle Interpolation* [Lag95] a également été utilisée.

Des analyses détaillées de la méthode peuvent se trouver dans [Mon92, Ben90b].

1.3.1 Construction

La méthode *SPH* découle de la théorie des distributions. Nous présentons la méthode et ses principaux résultats en donnant une esquisse de « construction » , inspirée de la théorie des distributions.

Considérons une fonction test f, infiniment continue, $f \in \mathcal{C}^\infty(\Omega)$ et à support compact sur Ω. Dans notre contexte de création 3D, le domaine considéré sera toujours implicitement un sous-espace compact de $\mathbb{R}^3$ représentant le volume dans lequel l'utilisateur peut travailler.

La distribution δ de Dirac se définit par la relation

$$f(0) = \int_\Omega \delta f(\mathrm{x}) d\mathrm{x}.$$

L'opération de translation étant bien définie sur les distributions, on peut utiliser un abus de notation et écrire

$$f(\mathrm{r}) = \int_\Omega \delta(\mathrm{x}-\mathrm{r}) f(\mathrm{x})\, d\mathrm{x}.$$

La distribution de Dirac n'est pas représentable numériquement. L'idée est donc de considérer une autre famille de distributions, représentable numériquement, et tendant vers la distribution de Dirac.

On introduit donc la fonction $W(\cdot, h)$:

$$\begin{array}{rccc} W & : & \mathbb{R}^3 & \to & \mathbb{R} \\ & & \mathrm{x} & \mapsto & W(\mathrm{x}, h), \end{array}$$

où h caractérise la taille du support de la fonction.

Cette fonction, appelée fonction noyau, est supposée localement intégrable. On définit l'opérateur d'approximation $\langle\cdot\rangle$

$$\langle f(\mathrm{r})\rangle = \int_\Omega W(\mathrm{x}-\mathrm{r}, h) f(\mathrm{x})\, d\mathrm{x}. \tag{1.3}$$

La convergence vers la distribution de Dirac s'écrit, pour toute fonction test f

$$\lim_{h\to 0} \langle f(\mathrm{r})\rangle = \lim_{h\to 0} \int_\Omega f(\mathrm{x}) W(\mathrm{x}-\mathrm{r}, h)\, d\mathrm{x} = \int_\Omega f(\mathrm{x})\delta(\mathrm{x}-\mathrm{r})\, d\mathrm{x} = f(\mathrm{r}).$$

Pour garantir la conservation des grandeurs physiques, on impose la condition de normalisation

$$\int_\Omega W(\mathrm{x}-\mathrm{r}, h)\, d\mathrm{x} = 1.$$

En faisant un développement de Taylor,

$$f(\mathrm{x}) = f(\mathrm{r}) + (\mathrm{x}-\mathrm{r})_i \frac{\partial}{\partial x_i} f(\mathrm{r}) + o(|\mathrm{x}-\mathrm{r}|),$$

on a

$$\begin{array}{rcl} \langle f(\mathrm{r})\rangle & = & \displaystyle\int_\Omega W(\mathrm{x}-\mathrm{r}, h) \left[f(\mathrm{r}) + (\mathrm{x}-\mathrm{r})_i \frac{\partial}{\partial x_i} f(\mathrm{r}) + o(|\mathrm{x}-\mathrm{r}|) \right] d\mathrm{x} \\ & = & \displaystyle f(\mathrm{r}) \int_\Omega W(\mathrm{x}-\mathrm{r}, h)\, d\mathrm{x} + \frac{\partial}{\partial x_i} f(\mathrm{r}) \int_\Omega (\mathrm{x}-\mathrm{r})_i W(\mathrm{x}-\mathrm{r}, h)\, d\mathrm{x} + o(|\mathrm{x}-\mathrm{r}|^2) \\ & = & \displaystyle f(\mathrm{r}) + \tfrac{\partial}{\partial x_i} f(\mathrm{r}) \int_\Omega (\mathrm{x}-\mathrm{r})_i W(\mathrm{x}-\mathrm{r}, h)\, d\mathrm{x} + o(|\mathrm{x}-\mathrm{r}|^2). \end{array}$$

En imposant en plus que le noyau ne dépende que de la distance $|\mathrm{x}-\mathrm{r}|$, l'intégrande devient une fonction impaire sur un domaine « symétrique » et donc l'intégrale est nulle. L'approximation souhaitée est ainsi obtenue et notre opérateur $\langle\cdot\rangle$ devient précis à l'ordre 2. Dans la pratique on utilise donc des noyaux radiaux que l'on note

$W(|\mathrm{x}-\mathrm{r}|,h)$. En allant plus loin dans le développement de Taylor, il est possible de déterminer d'autres conditions sur le noyau pour obtenir une précision plus grande.

On peut montrer que l'opérateur $\langle\cdot\rangle$ commute avec les opérateurs différentiels « classiques » (∇, $\nabla\cdot$, $\nabla\times$, Δ etc.). Par exemple :

$$\begin{aligned}\langle\nabla f(\mathrm{r})\rangle &= \int_\Omega W(|\mathrm{x}-\mathrm{r}|,h)\nabla_\mathrm{x} f(\mathrm{x})\,d\mathrm{x}\\ &= [f(\mathrm{x})W(|\mathrm{x}-\mathrm{r}|,h)]_\Omega - \int_\Omega \nabla_\mathrm{x} W(|\mathrm{x}-\mathrm{r}|,h) f(\mathrm{x})\,d\mathrm{x}.\end{aligned}$$

W ayant un support compact, le premier terme s'annule. On a aussi

$$\nabla_\mathrm{x} W(|\mathrm{x}-\mathrm{r}|,h) = -\nabla_\mathrm{r} W(|\mathrm{x}-\mathrm{r}|,h),$$

d'où

$$\langle\nabla f(\mathrm{r})\rangle = \int_\Omega \nabla_\mathrm{r} W(|\mathrm{x}-\mathrm{r}|,h) f(\mathrm{x})\,d\mathrm{x} = \nabla_\mathrm{r}\int_\Omega W(|\mathrm{x}-\mathrm{r}|,h) f(\mathrm{x})\,d\mathrm{x},$$

soit,

$$\langle\nabla f(\mathrm{r})\rangle = \nabla\langle f(\mathrm{r})\rangle. \tag{1.4}$$

Supposons à présent la connaissance d'une fonction f en un nombre fini de points de l'espace *i.e.*

$$f(\mathrm{r}) \approx \sum_{i=1}^{n} f(\mathrm{r}_i)\delta(\mathrm{r}_i-\mathrm{r}).$$

En appliquant l'opérateur $\langle\cdot\rangle$, il vient

$$\langle f(\mathrm{r})\rangle = \int_\Omega \left(\sum_{i=1}^{n} f(\mathrm{r}_i)\delta(\mathrm{r}_i-\mathrm{r})\right) W(|\mathrm{x}-\mathrm{r}|,h)\,d\mathrm{x}.$$

L'élément $d\mathrm{x}$ représente un volume et peut donc s'écrire comme le rapport d'une masse m sur une masse volumique[2] ρ. L'écriture précédente peut alors se simplifier et fournir l'opérateur discret d'interpolation

$$[[f(\mathrm{r})]] = \sum_{i=1}^{n} \frac{m(\mathrm{r}_i)}{\rho(\mathrm{r}_i)} f(\mathrm{r}_i) W(|\mathrm{r}_i-\mathrm{r}|,h). \tag{1.5}$$

Il s'agit de la formule fondamentale de la méthode.

Le passage au discret induit une nouvelle erreur d'approximation. On a donc une première erreur introduite par l'opérateur $\langle\cdot\rangle$ et une seconde erreur introduite par l'opérateur $[[\cdot]]$.

La formule est généralement illustrée par le calcul, non moins fondamental, de la densité :

$$[[\rho(\mathrm{r})]] = \sum_{i=1}^{n} \frac{m(\mathrm{r}_i)}{\rho(\mathrm{r}_i)} \rho(\mathrm{r}_i) W(|\mathrm{r}_i-\mathrm{r}|,h) = \sum_{i=1}^{n} m(\mathrm{r}_i) W(|\mathrm{r}_i-\mathrm{r}|,h). \tag{1.6}$$

[2]Par abus de langage, on parlera, par la suite, de densité, ce qui ne sera pas gênant si on utilise comme densité de référence celle de l'eau, soit 1. Il s'agit en fait d'un barbarisme puisque le terme anglais signifiant masse volumique est *density*.

Afin de démontrer la nécessité de la contrainte de normalisation, on peut écrire

$$\int_\Omega \rho(\mathrm{r})\, d\mathrm{r} = \int_\Omega \sum_i m(\mathrm{r}_i) W(|\mathrm{r}_i - \mathrm{r}|, h)\, d\mathrm{r} = \sum_i m(\mathrm{r}_i) \int_\Omega W(|\mathrm{r}_i - \mathrm{r}|, h)\, d\mathrm{r} = \sum_i m(\mathrm{r}_i).$$

Il est donc nécessaire d'avoir un noyau normalisé pour assurer la conservation de masse.

L'intérêt est, comme pour l'opérateur $\langle\cdot\rangle$, que tout opérateur différentiel commute avec l'opérateur $[[\cdot]]$. La méthode *SPH* permet donc l'approximation de grandeurs comme la densité mais également l'approximation de grandeurs différentielles, à l'instar des méthodes de type différences finies, utilisées en approche eulérienne.

Le support du noyau W étant borné, il sera important de pouvoir déterminer les valeurs ou particules à prendre en compte pour limiter les calculs. C'est la contrepartie des simplifications formelles du modèle lagrangien : la topologie du système n'est pas fixe et il est nécessaire de pouvoir construire le voisinage efficacement, à chaque calcul. Nous reviendrons sur cette problématique fondamentale des approches lagrangiennes dans la section 1.4.

Nous utiliserons à présent la notation A_i pour exprimer la valeur de la fonction A à la particule i de position r_i ($A_i = A(\mathrm{r}_i)$) et l'indice j fera référence aux particules voisines d'une particule. On omettra également la notation de l'opérateur d'approximation discrète $[[\cdot]]$.

Monaghan [Mon92] donne deux règles d'or pour bien utiliser la méthode *SPH* :

- *première règle d'or* : lorsqu'il s'agit d'interpréter physiquement les résultats, utiliser un noyau de type gaussienne ($\frac{\exp(-\frac{x^2}{h})}{h\sqrt{\pi}}$). Les gaussiennes n'ont pas de support compact mais ont des bonnes propriétés mathématiques, notamment de continuité.
- *deuxième règle d'or* : introduire la densité dans les formules d'interpolation. On introduit de la sorte une symétrie dans les formules ce qui apportera un gain de stabilité et également la conservation de certains principes physiques. Par exemple on pourra écrire

$$\nabla(\rho A) = \rho \nabla A + A \nabla \rho \Leftrightarrow \nabla A = \frac{1}{\rho} \left(\nabla(\rho A) - A \nabla \rho \right).$$

En appliquant les formules précédentes, il vient

$$\begin{aligned} \nabla A_i &= \frac{1}{\rho_i} \left(\left(\sum_j \frac{m_j}{\rho_j} (\rho_j A_j) \nabla W(\mathrm{r}_i - \mathrm{r}_j, h) \right) + \left(A_i \sum_j \frac{m_j}{\rho_j} \rho_j \nabla W(\mathrm{r}_i - \mathrm{r}_j, h) \right) \right) \\ &= \frac{1}{\rho_i} \sum_j m_j (A_j - A_i) \nabla W(\mathrm{r}_i - \mathrm{r}_j, h). \end{aligned}$$

Revenons à présent sur la première règle d'or et le choix du noyau, au coeur de la méthode elle-même.

1.3.2 Noyau

Le choix du noyau est très important puisqu'il intervient dans un grand nombre de calculs. Il est nécessaire de trouver un compromis entre le temps de calcul et la

précision.

Même si Monaghan préconise l'utilisation de gaussiennes pour les interprétations théoriques, elles ne sont guère utilisables dans la pratique. D'une part le calcul d'une exponentielle est très coûteux et d'autre part, le support de l'exponentielle n'est pas borné. Cela impliquerait d'effectuer les calculs sur l'ensemble de l'espace et non dans un voisinage.

Pour pallier ces problèmes, il est possible d'utiliser une *spline* imitant la courbure des gaussiennes [GM77, ML85] comme, par exemple (fig. 1.2)

$$W(\mathrm{r},h) = \frac{1}{\pi h^3} \begin{cases} 1 - \frac{3}{2}\left(\frac{||\mathrm{r}||}{h}\right)^2 + \frac{3}{4}\left(\frac{||\mathrm{r}||}{h}\right)^3 & \text{si} \quad 0 \leq ||\mathrm{r}|| \leq h \\ \frac{1}{4}\left(2 - \frac{||\mathrm{r}||}{h}\right)^3 & \text{si} \quad h < ||\mathrm{r}|| \leq 2h \\ 0 & \text{sinon.} \end{cases} \tag{1.7}$$

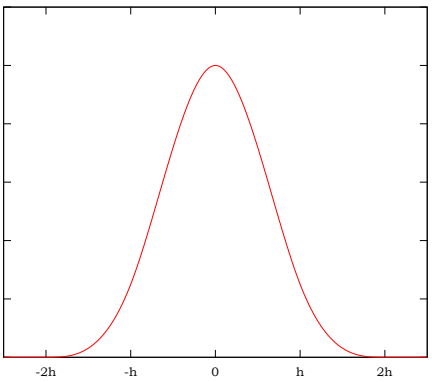


FIG. 1.2: Noyau polynomial de type gaussienne.

L'évaluation d'un polynôme de faible degré est rapide et, la définition par morceaux permet de contrôler la taille du support du noyau.

Le gradient de ce noyau s'annule en l'origine. Dans un contexte physique, cela se traduira par l'annulation de forces pour des particules proches. Ce phénomène pouvant être indésirable, il peut être intéressant d'utiliser un noyau continu mais non dérivable à l'origine.

Le noyau « *spiky* » , introduit par Desbrun [Des97], répond au problème :

$$W(\mathrm{r},h) = \begin{cases} \frac{15}{2\pi h^3}\left(1 - \frac{||\mathrm{r}||}{h}\right)^2 & \text{si } ||\mathrm{r}|| < h \\ 0 & \text{sinon.} \end{cases} \tag{1.8}$$

Son gradient est en effet non défini à l'origine :

$$\nabla W(\mathrm{r},h) = \begin{cases} -\frac{15}{\pi h^4}\left(1 - \frac{||\mathrm{r}||}{h}\right)\frac{\mathrm{r}}{||\mathrm{r}||} & \text{si } ||\mathrm{r}|| < h \\ 0 & \text{sinon.} \end{cases} \tag{1.9}$$

Le choix de la taille h du support est aussi très important. Un grand support implique un nombre important de particules à prendre en compte donc des résultats meilleurs mais au prix d'un temps de calcul plus important. Un petit support donne

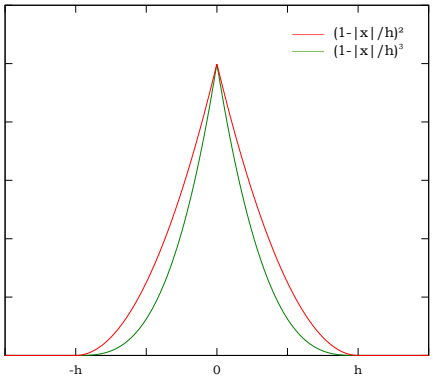


FIG. 1.3: Noyau *spiky*.

des résultats rapides mais moins précis. Krištof *et al.* notent que, pour obtenir une simulation stable, le support h doit être de taille

$$h \geq 2\Delta s,$$

où Δs représente l'espacement entre les particules [KBKv09].

Nous proposons d'évaluer la qualité des noyaux présentés précédemment en fonction de leur taille de support sur une configuration de référence (fig. 1.4). Ce test permet de mettre en évidence le filtrage effectué sur les particules isolées et sur les regroupements de particules (au niveau de l'anse), et également de montrer l'influence de l'interpénétration de particules. Les figures 1.5 et 1.6 présentent les densités obtenues respectivement avec le noyau de type gaussienne et avec le noyau *spiky*.

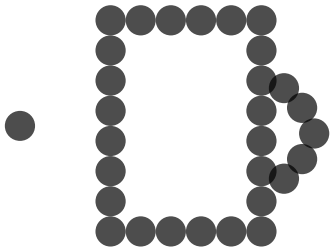
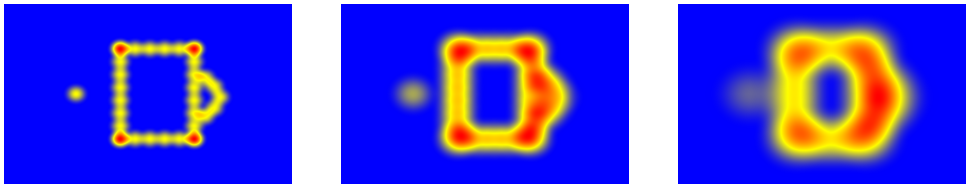


FIG. 1.4: Configuration de test de densité.

On constate notamment que le noyau de type gaussienne est beaucoup plus sensible à la répartition de la matière (fig. 1.7) : la particule isolée engendre un champ de densité très diffus. À l'inverse, les particules en coin et les particules s'interpénétrant, qui ont les voisinages de particules les plus grands, présentent des densités fortes.

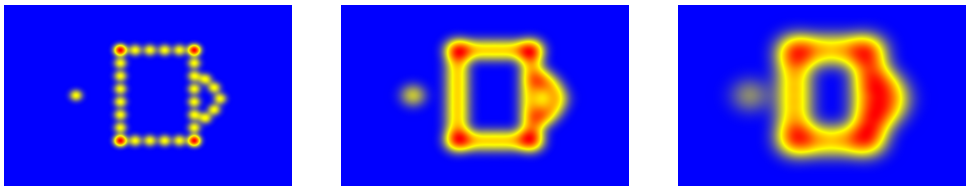
Le noyau *spiky* génère visuellement un champ de densité légèrement moins sensible à la répartition des particules. L'utilisation d'un support de taille $2\Delta s$ fournit des résultats lisses en conservant une densité faible au niveau du vide de l'arrondi.

La contrainte de simulation temps réel implique que nous utiliserons le moins de particules possible. L'utilisation d'un noyau assez peu sensible à la quantité de



(a) $h = \Delta s$ (b) $h = 2\Delta s$ (c) $h = 3\Delta s$

FIG. 1.5: Influence de la taille de support sur le calcul de densité par un noyau de type gaussienne.



(a) $h = \Delta s$ (b) $h = 2\Delta s$ (c) $h = 3\Delta s$

FIG. 1.6: Influence de la taille de support sur le calcul de densité pour le noyau *spiky*.

(a) $h = \Delta s$ (b) $h = 2\Delta s$ (c) $h = 3\Delta s$

FIG. 1.7: Différence des densités obtenues avec un noyau de type gaussienne et le noyau *spiky*.

matière est alors souhaitable pour obtenir des résultats corrects avec le moins de données possible. Le noyau *spiky* a donc été retenu avec un support de taille $h = 2\Delta s$ afin d'obtenir une bonne stabilité pour le coût le plus faible possible.

Nous avons à présent tous les éléments nécessaires pour utiliser la méthode d'approximation *SPH*. L'approximation calcule en quelque sorte une moyenne des grandeurs physiques locales.

Pour être pleinement efficaces, les calculs ne doivent reposer que sur les particules situées dans le support du noyau. Il est donc nécessaire de savoir rechercher le voisinage d'un point efficacement.

1.4 Recherche de voisinage

Les méthodes eulériennes nécessitent une discrétisation de l'espace aux noeuds desquels sont effectués les calculs et imposent une limitation de l'espace de simulation. Cependant, la topologie de la discrétisation est connue et figée ; chaque sommet

a généralement un ou deux voisins, successeur ou prédécesseur, dans chaque dimension.

Les approches lagrangiennes, modélisant les matériaux eux-mêmes, permettent une souplesse au niveau du domaine spatial de simulation. Néanmoins, les particules de la discrétisation ne sont pas fortement liées et les informations de voisinage ne sont pas figées. Nous avons vu que la méthode *SPH* permet l'approximation de grandeurs en approche lagrangienne, en ne prenant en compte que des valeurs situées dans le support d'un noyau lissant. Nous avons donc besoin de pouvoir déterminer la topologie locale en tout point ou, autrement dit, le voisinage de chaque particule.

Le voisinage peut se définir de plusieurs façons. On peut, par exemple, considérer que le voisinage est l'ensemble des particules situées à une certaine distance de la particule courante. On peut également considérer un voisinage défini par un nombre fixe de particules. Cette définition en nombre de particules présente le désavantage important de rendre la relation de voisinage non symétrique.

Par ailleurs, Wroblewski *et al.* [WKB07] ont montré que, dans un contexte de simulation de fluide incompressible, l'utilisation d'un voisinage défini par une distance fixe est plus efficace en temps de calcul pour des résultats visuellement similaires. Nous utiliserons donc un voisinage défini par une distance fixe. La recherche de voisinage est ainsi, dans notre cas, équivalente à la recherche d'un ensemble de points situés dans une boule de rayon h pour la distance euclidienne dans $\mathbb{R}^3$.

Ceci constitue le point central de l'algorithmique pour les systèmes de particules libres. L'approche naïve qui consiste à calculer, pour chaque point, la distance de celui-ci à l'ensemble des particules n'est, de part sa complexité linéaire en temps, pas satisfaisante si l'on veut traiter des systèmes avec un nombre important de particules.

Nous allons donc présenter les structures spatiales les plus fréquentes et détailler leur utilisation pour notre problématique de recherche de voisinage. Une comparaison sur des scénarios concrets nous permettra enfin de déterminer l'approche la plus efficace.

1.4.1 Grille

Le défaut de l'approche naïve réside dans l'aspect global de la recherche, *i.e.* l'ensemble du volume est pris en compte alors que le résultat recherché est local. Une première idée consiste à discrétiser l'espace selon une grille régulière sur chaque axe. Chaque particule est alors associée à un, et un seul, volume élémentaire ou *voxel* de la grille (fig. 1.8).

Le choix crucial devient ainsi le choix de découpe en voxels, pour lequel il existe au moins deux possibilités :

1. l'utilisation d'une grille finie paramétrée par la taille d'un voxel ;
2. l'utilisation d'une grille finie paramétrée par le nombre de voxels.

La deuxième solution peut être écartée rapidement. Cette solution permet de contrôler le coût mémoire de la structure efficacement. Néanmoins la solution ne se prête pas aux simulations effectuées sur un grand domaine. Les voxels obtenus pourraient contenir un grand nombre de particules et on augmenterait alors le nombre de tests à effectuer, dont beaucoup seraient inutiles.

Il reste à décider d'un découpage « relatif » de l'espace *i.e.* considérer le volume parallélépipédique occupé par les particules et le découper en voxel ou d'un

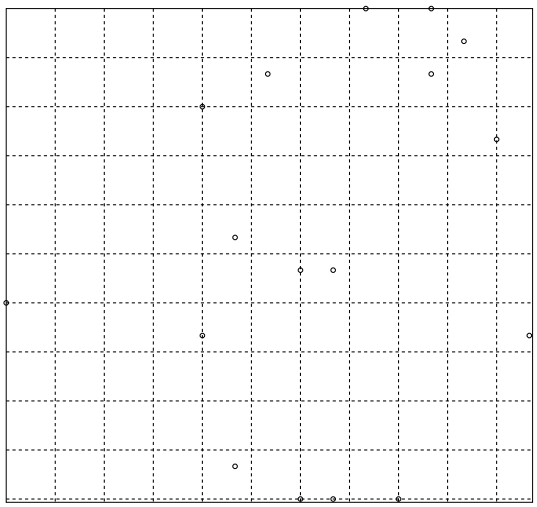


FIG. 1.8: Grille construite sur un nuage de points aléatoires.

découpage absolu *i.e.* effectuer dans un référentiel extérieur au système, *e.g.* dans le repère monde. La première solution a l'avantage d'utiliser moins de mémoire que la seconde et de ne pas contraindre explicitement le volume de travail de l'utilisateur. Cependant, dès que la boîte englobant la matière change de dimensions, il faut reconstruire entièrement la structure alors qu'un découpage absolu permet une mise à jour locale des voxels. Même si le découpage absolu nécessite plus de mémoire et contraint *de facto* l'utilisateur à travailler dans un volume prédéfini, il permet une mise à jour plus efficace.

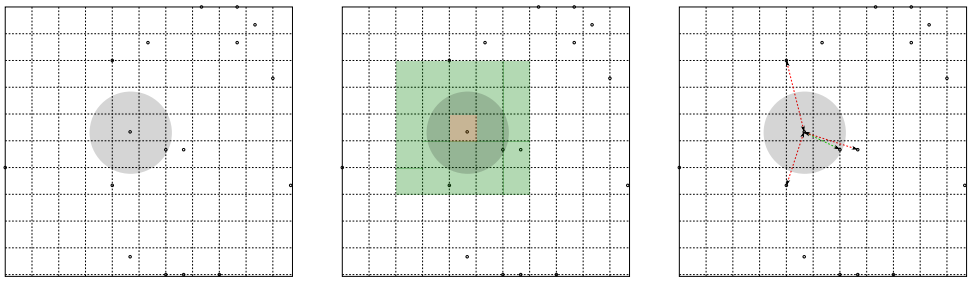
La recherche de voisinage s'effectue en déterminant le voxel contenant le point dont le voisinage est recherché, puis l'ensemble des voxels voisins à parcourir. Pour chacun de ces voxels, la distance du point à l'ensemble des particules contenues est calculée pour ne garder que les particules effectivement situées au plus à la distance de recherche (fig. 1.9).

Notons qu'une optimisation générale bien connue consiste à comparer les carrés des distances pour économiser un nombre important d'évaluations de racines carrées n'apportant pas d'information et coûteuses en temps de calcul.

La complexité d'une recherche de voisinage avec cette approche est constante en temps ; le nombre de particules contenues dans chaque voxel peut être majoré (en supposant que les particules ne s'interpénètrent pas) et le nombre de voxels à visiter aussi. Cependant la constante de complexité peut être mauvaise et va dépendre en grande partie du choix de la taille des voxels.
En effet ce choix déterminera le nombre maximal de particules par voxel (ratio du volume du voxel par le volume minimal d'une particule) et il aura une influence aussi sur le nombre de voxels à visiter. En notant d le rayon considéré pour la recherche des particules et h la taille des voxels, la taille k du voisinage de voxels à considérer est

$$k = \left\lfloor \frac{d}{h} \right\rfloor + 1.$$

Le meilleur choix de taille de voxels est donc $h = d$ puisqu'on a alors $k = 1$. On limite ainsi le parcours des voxels au 1-voisinage, soit 27 voxels. En fixant $h < d$, on doit parcourir un voisinage plus important, et en fixant $h > d$, on doit toujours parcourir



(a) Voisinage recherché. (b) Voxels à tester. (c) Tests effectués.

FIG. 1.9: Recherche de voisinage avec une grille.

le 1-voisinage mais chaque voxel contient plus de particules donc cela engendrerait un surplus de tests inutiles.

Dans le cas où le rayon de recherche d dépend des particules, on a intérêt à prendre pour taille le minimum des tailles de voisinages. On a alors toujours de bons résultats pour les particules ayant ce voisinage et pour les particules qui nécessitent un calcul de voisinage plus important, on parcourt plus de voxels mais chaque voxel contient moins de particules.

Müller *et al.* [MCG03] et Vesterlund [Ves04] utilisent une structure de grille pour la recherche de voisinage pour de la simulation de fluide.

1.4.2 Table de hachage

Le principal inconvénient lié à l'utilisation d'une grille est la surconsommation mémoire. En effet, pour une simulation donnée, toutes les cases de la grille sont allouées. En plus de la consommation mémoire excessive, l'utilisation d'une grille impose de définir une résolution fixe ou, à défaut, nécessite une ré-allocation mémoire coûteuse. Ceci entraîne une contrainte potentiellement forte puisque cela revient à considérer qu'on travaille dans une zone figée de l'espace.

Kipfer *et al.* [KSW04, KW06] ont introduit la structure de *staggered grid* ou grille échelonnée. Ils proposent d'associer à chaque particule un identifiant sur 64 bits de la forme $\mathrm{id} = |0|\mathrm{id}_x|\mathrm{id}_y|\mathrm{id}_z|$. Les auteurs proposent de trier la liste des particules en fonction de chaque partie de leur identifiant de voxel ; en utilisant ce tri, il suffit, pour chaque particule, de parcourir la liste jusqu'à trouver un identifiant de particule

$$\mathrm{id}_x \leq \mathrm{id}_x^{self} - 2,$$

auquel cas on a quitté le 1-voisinage.

On traite les paires de particules une seule et unique fois en traitant la particule dont l'identifiant de voxel est le plus grand. A l'intérieur d'un voxel, on ne traite que la première particule du voxel. On utilise deux autres listes triées sur les identifiants en id_y et id_z pour finir la recherche en utilisant le premier bit pour ne pas considérer plusieurs fois une même paire lors du traitement des différentes directions. L'approche permet le stockage compact voulu mais nécessite trois opérations de tri. Les auteurs proposent d'utiliser l'implémentation `std::sort` fournit par la `STL` et dont la complexité moyenne en temps est en $\mathcal{O}(n \log n)$. De plus, l'approche ne permet pas de traiter l'ensemble des paires puisqu'on ne regarde pas le 1-voisinage exact mais uniquement les voxels voisins selon les directions principales. Les voxels

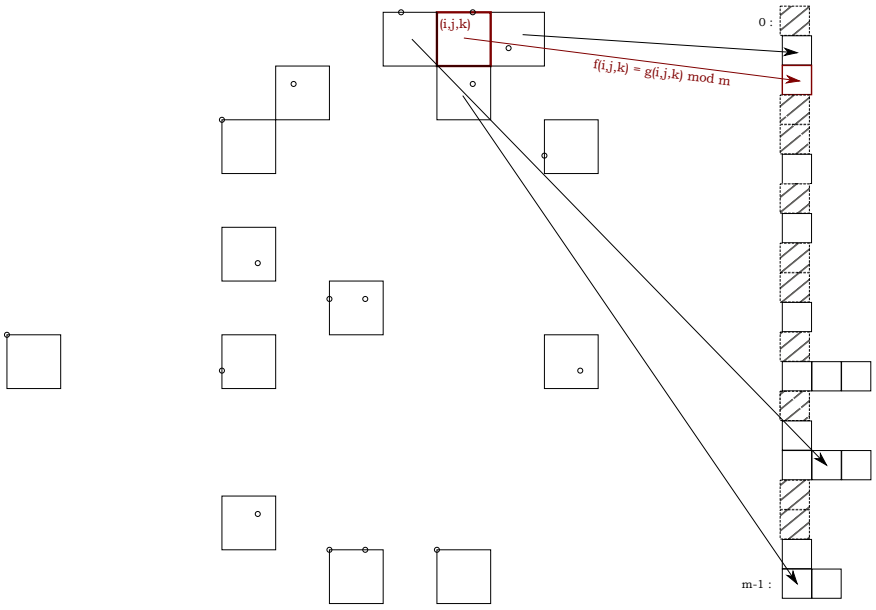


FIG. 1.10: Stockage d'une table de hachage.

voisins en diagonale ne sont pas traités.

Les tables de hachage permettent de lever ces problèmes en n'allouant les voxels d'une grille qu'aux endroits nécessaires. Une table de hachage permet de stocker des couples (clé, valeur) dans un tableau unidimensionnel. Dans notre contexte, nous allons ainsi considérer la valeur comme étant l'ensemble des particules contenues dans un voxel. Un voxel sera identifié par un triplet d'indices entiers (i, j, k) unique déterminant sa position dans l'espace et constituant sa clé (fig. 1.10).

La position dans la table de hachage est obtenue en deux étapes : le hachage à proprement parler, générant un entier à partir du triplet donné puis le modulo de ce nombre à la taille m du tableau utilisé pour le stockage dans la table.

L'injectivité de la fonction de hachage n'est pas supposée. L'opération de modulo peut donc attribuer la même position à deux clés distinctes. On parle alors de *collision*. Il y a collision lorsque deux clés distinctes ont la même valeur de hachage ou la même valeur après prise en compte du modulo sur le hachage. Pour gérer la possibilité des collisions, nous utilisons, pour chaque valeur de hachage, une liste chaînée contenant la clé d'origine *i.e.* le triplet (i, j, k) associé au voxel, et la liste des particules contenues dans le voxel correspondant. Il est cependant nécessaire de chercher à minimiser le nombre de collisions pour obtenir les meilleures performances.

Le choix de la fonction de hachage,

$$f \ : \ \mathbb{Z}^3 \to [[0; m-1]], f = h \circ g \text{ avec } g : \mathbb{Z}^3 \to \mathbb{N} \text{ et } h \equiv \cdot \bmod m,$$

détermine la position du stockage du couple (clé, valeur) dans la table. Il est crucial pour obtenir de bonnes performances.

La fonction doit être simple et rapide à évaluer et doit distribuer le mieux possible les triplets sur les entiers pour minimiser les risques de collision. Il est aussi préférable d'obtenir des hachages proches pour des triplets proches afin de tirer parti au mieux

du *cache* du processeur. Néanmoins, le contrôle du nombre de collisions est plus important pour obtenir de bonnes performances.

L'opération de modulo, qui fonctionne par intervalles de taille m, permet de conserver globalement les propriétés de la fonction de hachage.

Une bijection de $\mathbb{Z}^3 \to \mathbb{N}$ peut être utilisée comme fonction de hachage. Une telle bijection peut s'obtenir en composant des bijections « simples ». En introduisant les bijections b et c définies par

$$\begin{array}{rccc} b : & \mathbb{N}^2 & \to & \mathbb{N} \\ & (i,j) & \mapsto & b(i,j) = \frac{(i+j)(i+j+1)}{2} + j, \end{array}$$

et

$$\begin{array}{rccc} c : & \mathbb{Z} & \to & \mathbb{N} \\ & i & \mapsto & c(i) = \begin{cases} 2i & \text{si} \quad i \geq 0 \\ 2i+1 & \text{si} \quad i < 0, \end{cases} \end{array}$$

on obtient la bijection voulue par

$$\begin{array}{rccc} g : & \mathbb{Z}^3 & \to & \mathbb{N} \\ & (i,j,k) & \mapsto & g(i,j,k) = b\Big(b\big(c(i), c(j)\big), c(k)\Big). \end{array}$$

Cependant, l'évaluation d'une telle fonction est coûteuse. Par ailleurs, comme les données sont non déterminées à l'avance et dynamiques, il n'est pas nécessaire de prétendre rechercher un hachage parfait *i.e.* sans collision, contrairement aux propositions de Lefebvre et Hoppe [LH06] pour des données statiques.

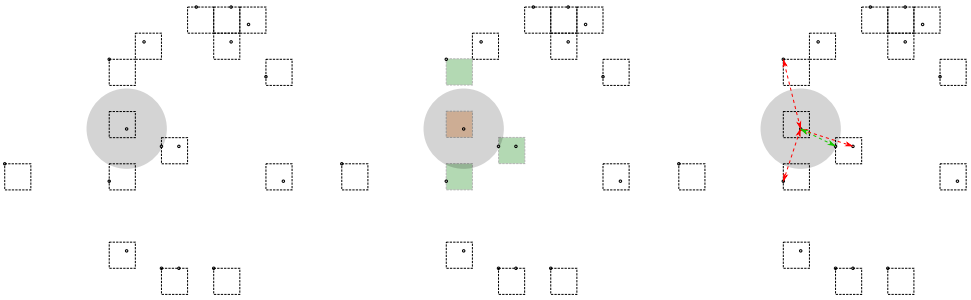
Teschner *et al.* [THM+03] ont utilisé des tables de hachage pour accélérer la détection des collisions d'une scène. Ils proposent d'utiliser la fonction de hachage suivante

$$\text{hash}(i,j,k) = (i \cdot p_1 \,\mathbf{xor}\, j \cdot p_2 \,\mathbf{xor}\, k \cdot p_3) \mod n,$$

où n est la taille de la table et $p_1 = 73856093$, $p_2 = 19349663$ et $p_3 = 83492791$.

Des tests nous ont permis de constater que la fonction de hachage `lookup2` [Jen97] donnaient de meilleurs résultats pour nos problèmes. L'idée n'est plus de considérer explicitement les triplets en entrée comme des valeurs entières mais d'utiliser leur représentation binaire (typiquement 32 bits). Il s'agit alors de mélanger les informations binaires des trois valeurs en entrée pour distribuer au mieux la présence de 0 et de 1 dans la représentation binaire de sortie. L'écriture de la fonction de hachage en C est :

```
int lookup2 (int i, int j, int k)
{
    i -= j; i -= k; i ^= (k>>13);
    j -= k; j -= i; j ^= (i<<8);
    k -= i; k -= j; k ^= (j>>13);
    i -= j; i -= k; i ^= (k>>12);
    j -= k; j -= i; j ^= (i<<16);
    k -= i; k -= j; k ^= (j>>5);
    i -= j; i -= k; i ^= (k>>3);
    j -= k; j -= i; j ^= (i<<10);
    k -= i; k -= j; k ^= (j>>15);
    return k;
}
```



(a) Voisinage recherché. (b) Voxels à tester. (c) Tests effectués.

FIG. 1.11: Recherche de voisinage dans une table de hachage.

Les *Arithmetic and Logical Unit* (*ALU*) des processeurs modernes étant rapides pour les opérations binaires, la fonction est très peu coûteuse.

La recherche de voisinage s'effectue en déterminant le voxel contenant la particule et en calculant le k-voisinage de voxels à considérer. La taille k de voisinage de voxels à considérer est obtenu, comme dans le cas des grilles, par

$$k = \left\lfloor \frac{d}{h} \right\rfloor + 1.$$

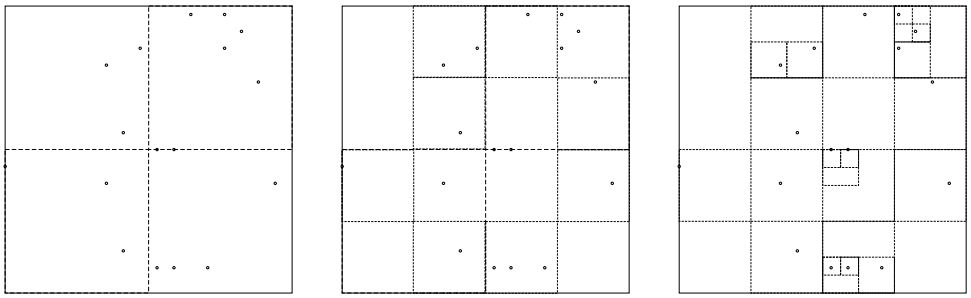
Ceci nous amène donc à considérer des voxels de dimensions $h = d$ pour limiter la recherche du voisinage d'une particule au 1-voisinage de voxels. Chaque triplet du 1-voisinage de voxels à considérer est ensuite haché et, pour chaque voxel existant dans la table, un calcul exhaustif de la distance de la particule courante aux particules contenues dans le voxel est effectué (fig. 1.11). Le nombre de candidats est donc exactement le même que pour une structure de grille mais la mise à jour est plus efficace. Celle-ci n'est pas liée au volume occupé par les particules, et la structure est plus compacte en mémoire.

La structure de table de hachage pour la recherche de voisinage a notamment été utilisée par Clavet *et al.* [CBP05].

1.4.3 Octree

Plutôt que de découper l'espace régulièrement et de ne stocker que les zones contenant des particules, il est également possible d'utiliser des structures arborescentes prenant en compte l'agencement de la matière pour la découpe spatiale.

L'octree, qui peut se définir comme une grille adaptative, fait partie des structures arborescentes les plus utilisées. Le découpage d'un octree est régulier : on part d'une boîte parallélépipédique (généralement un cube) englobante divisée en son milieu selon les trois dimensions canoniques, soit en 8 fils. Le processus est conduit récursivement sur chacun des fils créés tant que le fils n'est pas vide et qu'aucun critère d'arrêt n'est rencontré (fig. 1.12). Ces critères d'arrêt peuvent être une profondeur d'arbre maximale et/ou le nombre d'éléments contenus dans un fils. Dans notre contexte, il est possible de majorer le nombre d'éléments contenus dans un volume donné ; nous avons donc opté pour l'utilisation d'un critère sur le nombre d'éléments seulement qui permet indirectement de controler la profondeur maximale de l'arbre.



(a) Premier niveau. (b) Deuxième niveau. (c) Octree final.

FIG. 1.12: Construction d'un octree.

La recherche de voisinage est également effectuée récursivement. On utilise deux étapes, imitant le principe de *broad phase*, filtrage rapide des morceaux d'espace à ne pas traiter, et *narrow phase*, traitement exhaustif des voxels candidats retenus, des méthodes de recherche de collisions entre objets.

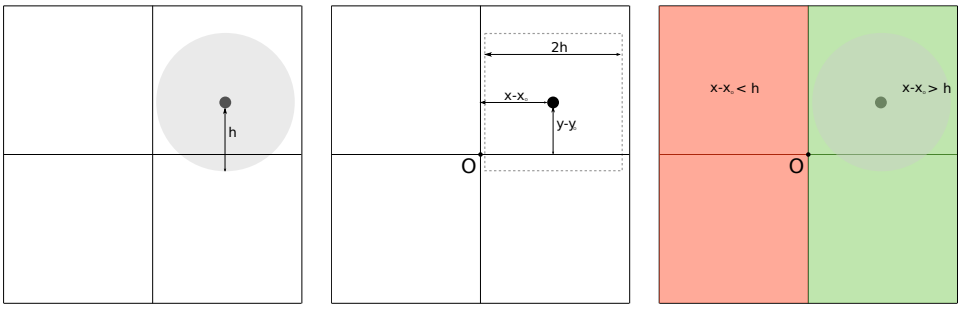
Il est possible de déterminer les branches de l'arbre à parcourir en calculant la distance des voxels à la boule représentant le volume de recherche. Le filtrage sur les voxels est alors précis. Cependant, le calcul de l'intersection boule-voxel est coûteux. Nous avons donc opté pour une stratégie de *broad phase* légèrement moins précise mais beaucoup plus rapide.

Nous utilisons le fait que le découpage de l'arbre est aligné sur les axes canoniques et donc tous les plans considérés sont parallèles. La découpe d'un voxel en 8 sous-voxels se fait par l'introduction de 3 plans, créant 6 demis espaces. Comme un voxel fils « hérite » des propriétés de son voxel parent, on peut considérer que chaque sous-voxel est l'intersection de 3 sous-espaces au lieu de 6. La boule de recherche peut également être approchée par le cube circonscrit et aligné sur les axes canoniques (fig. 1.13). Déterminer si ce cube intersecte chaque demi-espace peut s'effectuer au moyen d'une comparaison (fig. 1.13b). En utilisant les résultats de ces 6 comparaisons, on peut alors déterminer si le cube intersecte chaque sous-voxel avec une comparaison. Lorsque le voxel testé est une feuille, un test exhaustif des distances du point de recherche au contenu du voxel est effectué (fig. 1.14).

Le volume défini par le cube étant plus important que celui de la boule, il est possible d'explorer des voxels inutiles. Cependant, ces voxels inutiles peuvent être écartés rapidement à mesure que l'arbre est exploré en profondeur. Et, dans la mesure où les voxels feuille contiennent peu de particules, tester exhaustivement quelques voxels inutiles est peu coûteux (fig. 1.14).

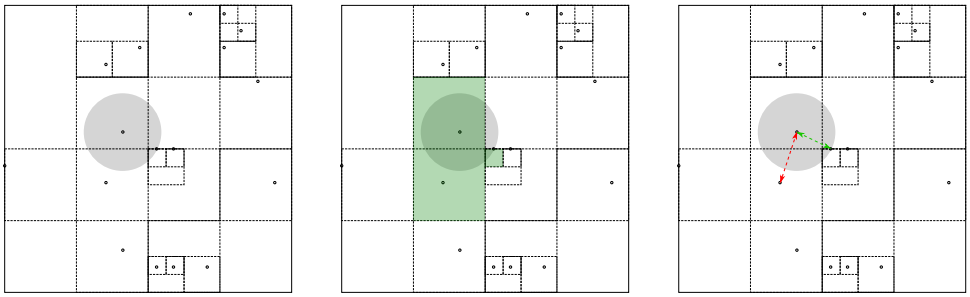
La mise à jour de la structure est délicate. Elle repose principalement sur le volume défini par le voxel racine. Tant que les particules sont situées à l'intérieur de celui-ci, il est facile de retirer une particule d'un voxel et de la réinsérer dans un autre. Si le voxel de destination est plein, alors on le redécoupe récursivement. Si une particule sort du volume, alors il est possible de considérer la racine comme voxel fils d'une racine supérieure.

Cependant, une telle mise à jour augmente la profondeur d'arbre inutilement. Nous avons préféré ne pas implémenter d'heuristique de mise à jour de la structure et reconstruire l'arbre intégralement afin de garantir que la structure corresponde au mieux au volume effectivement occupé par les particules.



(a) Recherche de voisinage dans l'arborescence de l'octree.
(b) Approximation de la boule de recherche par un cube.
(c) Filtrage obtenu.

FIG. 1.13: Méthode de filtrage utilisée pour la recherche dans octree.



(a) Voisinage considéré.
(b) *Broad phase.*
(c) *Narrow phase.*

FIG. 1.14: Recherche de voisinage dans un octree.

Desbrun [Des97] utilise une structure d'octree pour la recherche de voisinage dans un contexte de simulation de matériau hautement déformable.

1.4.4 *k*d-tree

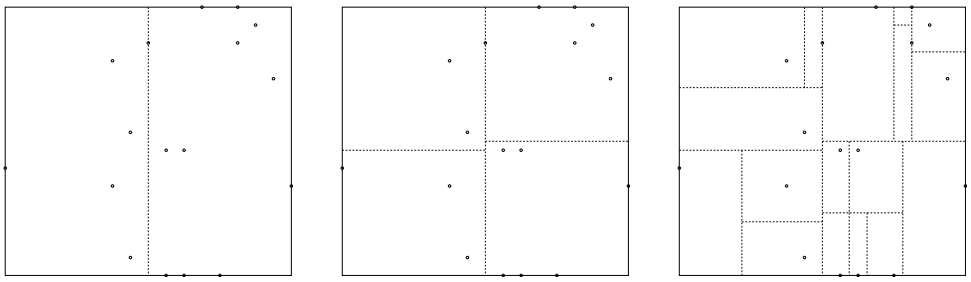
La structure d'octree est conceptuellement plus satisfaisante que l'approche par grille puisqu'elle concentre les efforts dans les régions non vides. Cependant, le découpage effectué est très régulier et peut ne pas être le plus adapté pour des configurations quelconques.

Le principe des *k*d-tree est de diviser récursivement les voxels selon un plan de coupe aligné avec les axes principaux :

- soit dans un ordre déterminé par avance, *i.e.* on définit un ordre sur les plans de coupe,
- soit calculé en fonction du contenu du voxel, *i.e.* en déterminant le meilleur plan de coup pour le sous-ensemble d'objets à représenter.

Le découpage est adaptatif et entièrement dynamique puisqu'il est fait « au mieux » et n'est pas prédéterminé comme dans le cas des octrees.

Ce découpage dynamique nécessite la détermination de plusieurs paramètres :



(a) Premier niveau. (b) Deuxième niveau. (c) kd-tree final.

FIG. 1.15: Construction de kd-tree.

- critère d'arrêt du découpage (comme pour les octrees, généralement profondeur de l'arbre et/ou nombre d'éléments contenus dans le voxel),
- choix de la dimension de la coupe,
- choix du placement de la coupe dans la dimension choisie.

Concernant le choix du critère d'arrêt, pour notre problématique, il semble là encore propice de fixer un critère sur le nombre de particules par voxels puisque lors de la recherche effective des voisins, nous aurons à calculer la distance d'une particule à toutes les particules des voxels candidats sélectionnés.

Les découpages sont effectués selon les directions canoniques du repère. Un choix naturel pour la dimension de coupe consiste à calculer la boîte englobante de l'ensemble des données et à prendre la dimension dans laquelle la boîte est la plus grande. De cette façon, on tend à avoir les voxels les plus réguliers possibles.

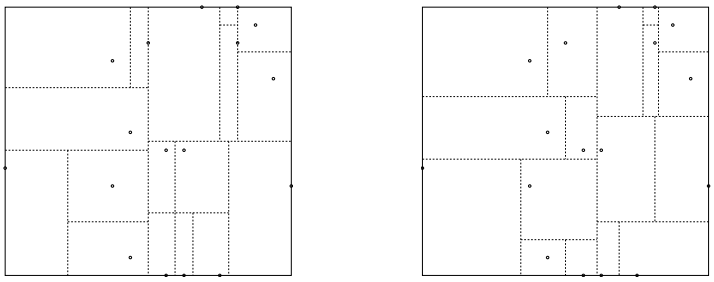
Plusieurs stratégies de placement du plan de coupe sont possibles :

- placement au milieu de la cellule pour avoir des voxels les plus réguliers possibles en terme de dimension,
- placement à la médiane de la distribution des particules pour avoir des voxels les plus réguliers possibles en terme de nombre de particules.

La détermination de la médiane nécessite un tri des données à chaque profondeur de l'arbre et induit un coût important. La solution de découpe au milieu de la plus grande dimension de la boîte englobante a donc été retenue (fig. 1.15).

Un inconvénient des kd-trees est la nécessité d'avoir la connaissance de toute l'information au moment de la construction. Comme chaque étape du découpage dépend fortement de l'information contenue, il n'est pas intéressant de vouloir mettre à jour un kd-tree.

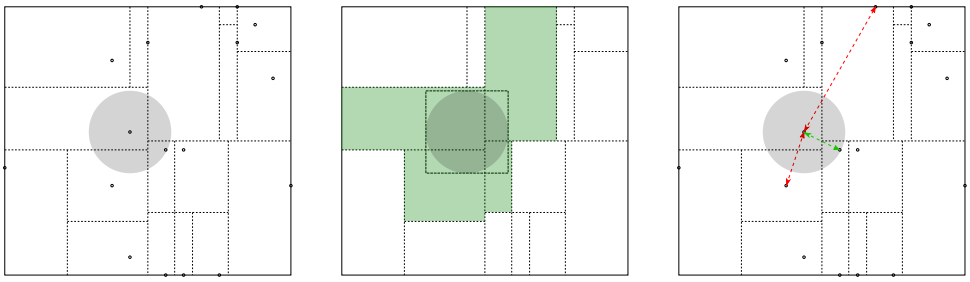
Notre implémentation pour la construction de l'arbre s'inspire librement de Bentley [Ben90a]. Cependant, nous avons besoin de connaître l'ensemble des voisins d'un point et non seulement plus proche voisin. Un test rapide permettant d'écarter les voxels trop éloignés est donc nécessaire. Arya et Mount [AM93] proposent un algorithme incrémental de calcul de la distance d'un point aux descendants d'un voxel. La division d'un voxel conserve la distance du voxel fils le plus proche et modifie la



(a) kd-tree avec découpe au milieu.

(b) kd-tree avec découpe à la médiane.

FIG. 1.16: Comparaison de k-trees.



(a) Voisinage considéré.

(b) *Broad phase.*

(c) *Narrow phase.*

FIG. 1.17: Recherche de voisinage avec un kd-tree.

distance à l'autre voxel fils d'un décalage dans la dimension du plan de coupe. La distance peut donc être mise à jour incrémentalement.

Néanmoins, cette stratégie reste relativement coûteuse. Par ailleurs, les plans de coupe sont alignés sur les axes canoniques. Le filtrage réalisé pour les octrees s'adapte également à la structure de kd-tree. Comme chaque voxel ne crée que deux demi-espaces, il suffit de deux comparaisons pour déterminer si les sous-voxels doivent être visités ou non (fig. 1.17).

À noter que le kd-tree est un cas particulier d'arbre de partitionnement binaire. Il est également possible d'utiliser des plans de coupe non alignés sur les axes canoniques. Cependant, la détermination de plan de coupe quelconque nécessite plus de calculs et ne permet plus notre stratégie de parcours rapide. Nous n'avons donc pas implémenté de tels arbres.

L'utilisation de kd-tree pour une simulation lagrangienne de fluide est par exemple utilisée par Shen et Shah [SS07] et Adams *et al.* [APKG07].

1.4.5 Hybridation

Jusqu'à présent, nous avons présenté des structures de découpage spatial de façon indépendante. Il semble intéressant de s'interroger sur la possibilité de coupler plusieurs structures entre elles pour utiliser les forces de chacune en espérant aussi minimiser les défauts de l'ensemble. Des présentations précédentes, nous pouvons faire une première analyse synthétisée dans la table 1.1.

	Avantages	**Inconvénients**
grille	• mise à jour efficace	• surconsommation mémoire • limitation de l'espace • nombre potentiellement élevé d'évaluations de distances
table de hachage	• mise à jour efficace, • stockage compact, • espace infini (aux limites numériques près)	• nombre potentiellement élevé d'évaluations de distances
octree	• adaptatif, • recherche efficace, • espace infini	• mise à jour peu efficace
kd-tree	• adaptatif, • recherche très efficace, • espace infini	• mise à jour non efficace

TAB. 1.1: Analyse des structures utilisées pour la recherche spatiale.

Havran *et al.* [HHS06] mentionnent deux approches multi-niveaux à base de kd-trees pour du rendu par *ray-tracing* sur des scènes non statiques :

1. un regroupement local de plusieurs objets dans un kd-tree, puis l'ensemble des kd-trees est regroupé dans un kd-tree ;

2. une structure pour les objets fixes, une structure pour les objets mouvants (dont le cardinal est typiquement plus faible que celui des objets fixes).

Cependant, au vu du tableau précédent, la structure de kd-tree n'est pas efficace en terme de mise à jour. Contrairement à l'octree qui est un découpage adaptatif mais régulier, le kd-tree s'adapte pour correspondre au mieux aux données qu'il contient. La recherche de voisinage est donc plus efficace. La table de hachage permet, elle, une mise à jour très efficace et la représentation d'un volume infini contrairement aux grilles. Une structure de table de hachage contenant des kd-trees pour représenter au mieux le volume de chaque voxel semble alors intéressante (fig. 1.18).

La construction est effectuée en répartissant l'ensemble des particules dans les voxels de la table de hachage puis, pour chaque voxel inséré, le kd-tree associé est construit.

La recherche est semblable à la recherche de chacune des structures, effectuées en cascade. On commence par déterminer l'ensemble des voxels voisins à parcourir puis on effectue une recherche sur les kd-trees associés à ces voxels.

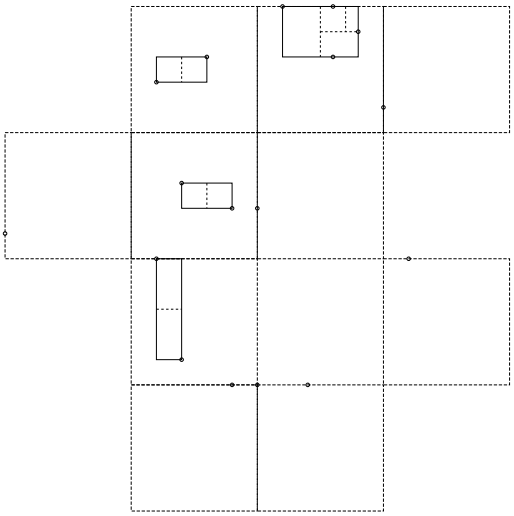


FIG. 1.18: Arbre hybride.

La mise à jour est relativement efficace et peut être qualifiée de « semi-locale » . Pour chaque voxel modifié, le kd-tree associé est reconstruit. La quantité d'information par voxel devant être relativement faible, la mise à jour devrait être efficace.

1.4.6 Comparatifs

Notre objectif final étant la sculpture de pâte à modeler, nous serons amenés à considérer deux problèmes distincts :

1. déplacement de l'ensemble particules, par exemple, par simulation de comportements physiques ;

2. déplacement d'un ensemble restreint de particules, par exemple, via l'imposition de contraintes locales par l'utilisateur.

Nous avons donc mis en place deux tests distincts.

Le premier test est une simulation des équations d'écoulement fluide sur l'ensemble des particules (fig. 1.19). La simulation nécessite de déterminer le voisinage de chaque particule, et chaque particule est ensuite déplacée ce qui nécessite donc une mise à jour complète de la structure de recherche. Les résultats présentent le temps moyen nécessaire à la mise à jour de la structure et le temps moyen passé à déterminer le voisinage de chaque particule pour différentes tailles de systèmes.

Les complexités asymptotiques théoriques sont vérifiées :

- $\mathcal{O}(n^2)$ pour la recherche naïve,
- $\mathcal{O}(n)$ pour la grille et la table de hachage
- $\mathcal{O}(n \log n)$ pour les structures d'arbre, octree et kd-tree.

On constate également que les structures sont proportionnellement principalement coûteuses en temps de recherche et non en temps de construction, hormis pour

la structure de kd-tree où le temps de construction représente le tiers du temps total pour des systèmes de moins de 20000 particules, puis prend jusqu'aux deux tiers du temps d'utilisation de la structure pour un système de 32768 particules. La structure hybride de table de hachage de kd-tree semble hériter principalement des propriétés de la structure de table de hachage. Ceci peut s'expliquer par la localité des kd-trees qui induit un coût quasi constant de traitement lorsque, comme ici, l'information est répartie de façon relativement homogène dans les kd-trees. Les cellules de la table de hachage étant de taille importante, la structure hybride est finalement peu adaptée à nos besoins.

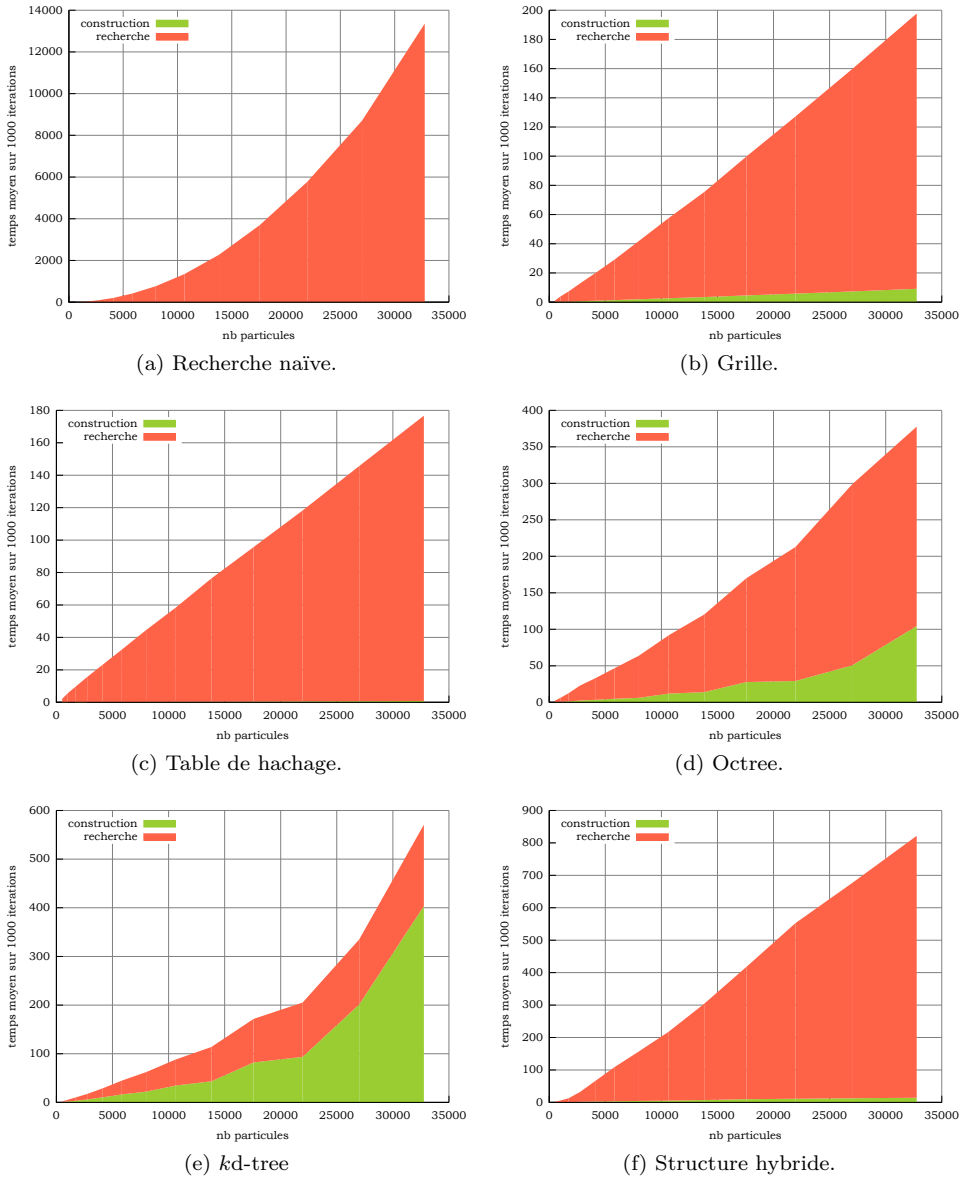
Le second test perturbe de façon aléatoire les particules en imitant l'action d'un outil sphérique qui pousserait les particules situées dans son volume, qui est supposé petit devant le volume total modélisé par les particules (fig. 1.20). La recherche de voisinage impacte un sous-ensemble des particules du système pour lesquelles il faut ensuite mettre à jour les données dans les structures de recherche. Ce test permet de tester principalement les temps de mise à jour et pénalise donc fortement les structures nécessitant une reconstruction complète. De plus, on effectue des recherches de voisinage de taille variable pour étudier la sensibilité des structures au rayon de recherche. On constate ici une dégradation des résultats obtenus avec les structures arborescentes lors de recherches avec un grand rayon. Ceci s'explique par le fait qu'une recherche avec un grand rayon va nécessiter le parcours d'une grande partie de l'arbre (voire de sa totalité) sans que cela apporte d'information. Ces structures paient donc le coût d'un parcours d'arbre et d'un calcul (quasi) exhaustif des distances des particules au centre de recherche.

La figure 1.21 permet de comparer dans le détail les résultats. Dans un souci de lisibilité, la moins bonne structure, à savoir la recherche naïve pour la simulation d'un écoulement fluide et le kd-tree pour les perturbations aléatoires locales, n'est pas présentée.

La comparaison de l'ensemble des structures permet de constater que la structure de table de hachage est la meilleure pour le premier scénario nécessitant une mise à jour et une recherche pour l'ensemble du système. Dans le second scénario, certaines recherches sont effectuées pour des grands rayons et impactent alors un nombre important de particules. La recherche naïve fournit ainsi les meilleurs résultats. Cependant, les temps de recherche pour la table de hachage sont relativement comparables.

Au final, la table de hachage donne de très bons résultats pour nos deux scénarios cibles. Nous avons donc retenu cette structure pour les recherches de voisinage.

Nous avons maintenant des outils performants pour évaluer des grandeurs dans l'espace. Il nous reste donc à définir un intégrateur afin de pouvoir simuler des comportements physiques dans le temps.



(a) Recherche naïve.

(b) Grille.

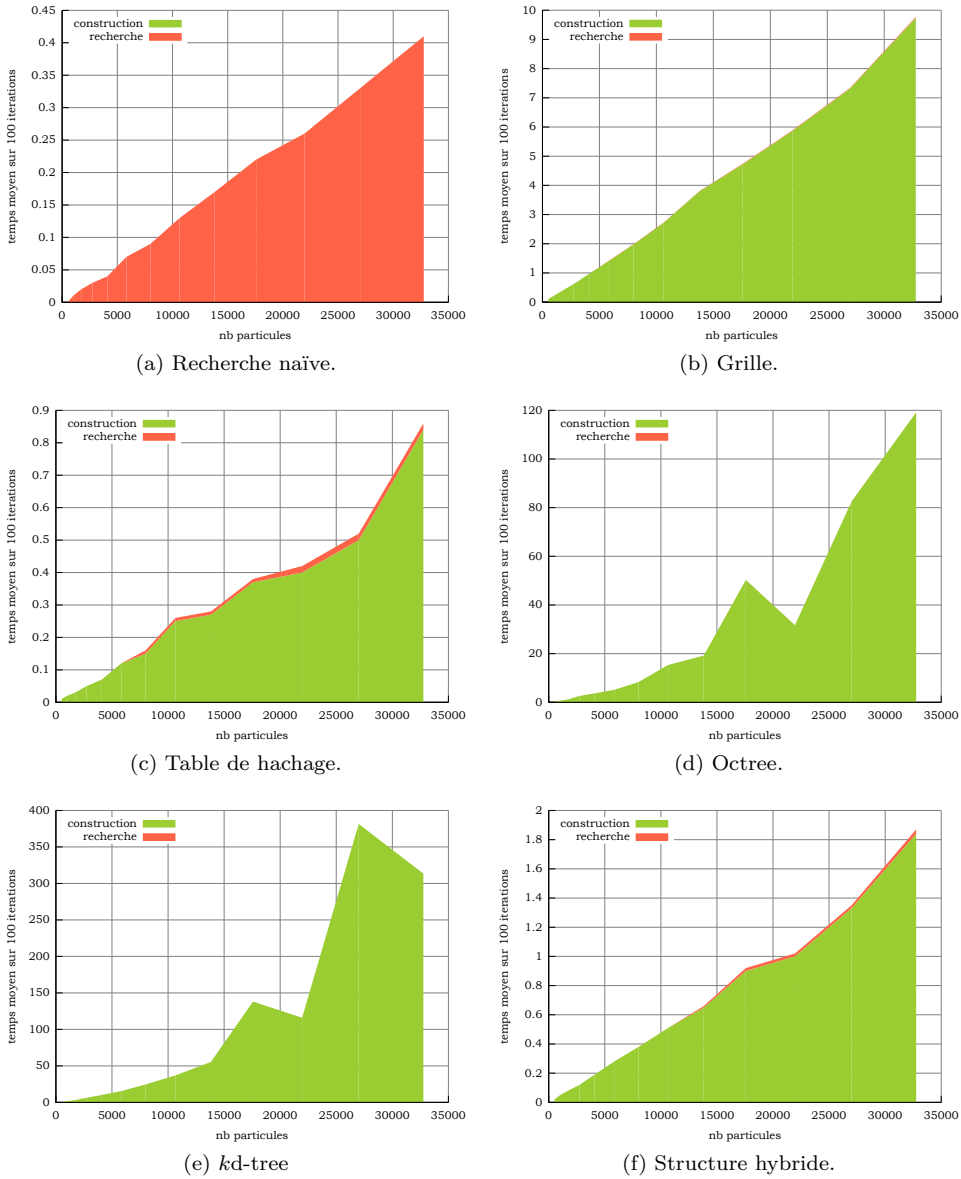
(c) Table de hachage.

(d) Octree.

(e) kd-tree

(f) Structure hybride.

FIG. 1.19: Comparaison des structures de recherche pour la simulation des équations de Navier-Stokes.



(a) Recherche naïve.

(b) Grille.

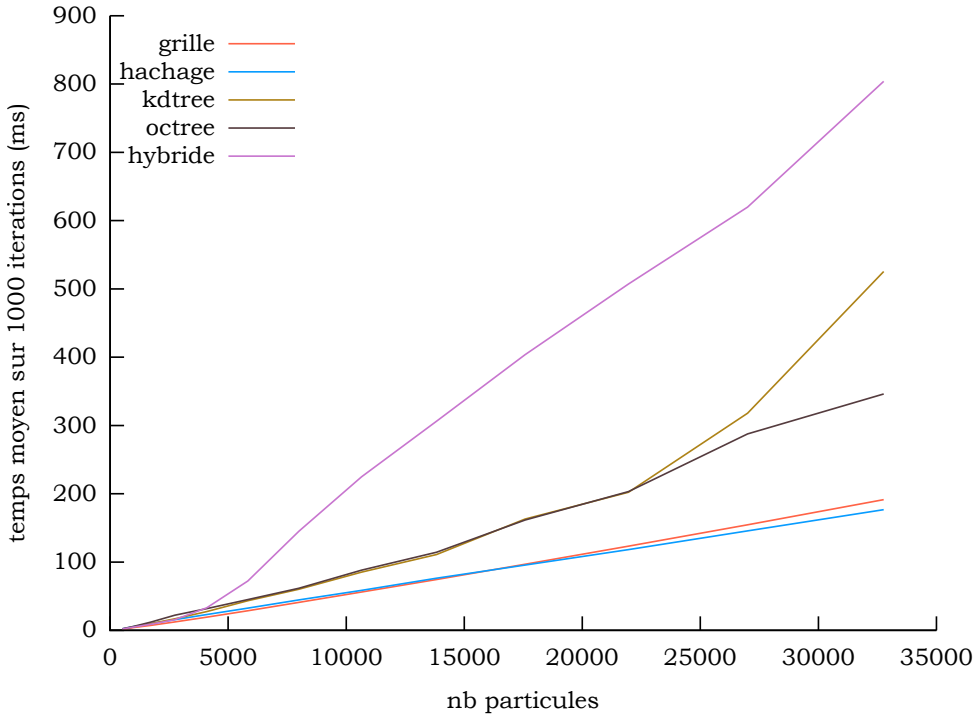
(c) Table de hachage.

(d) Octree.

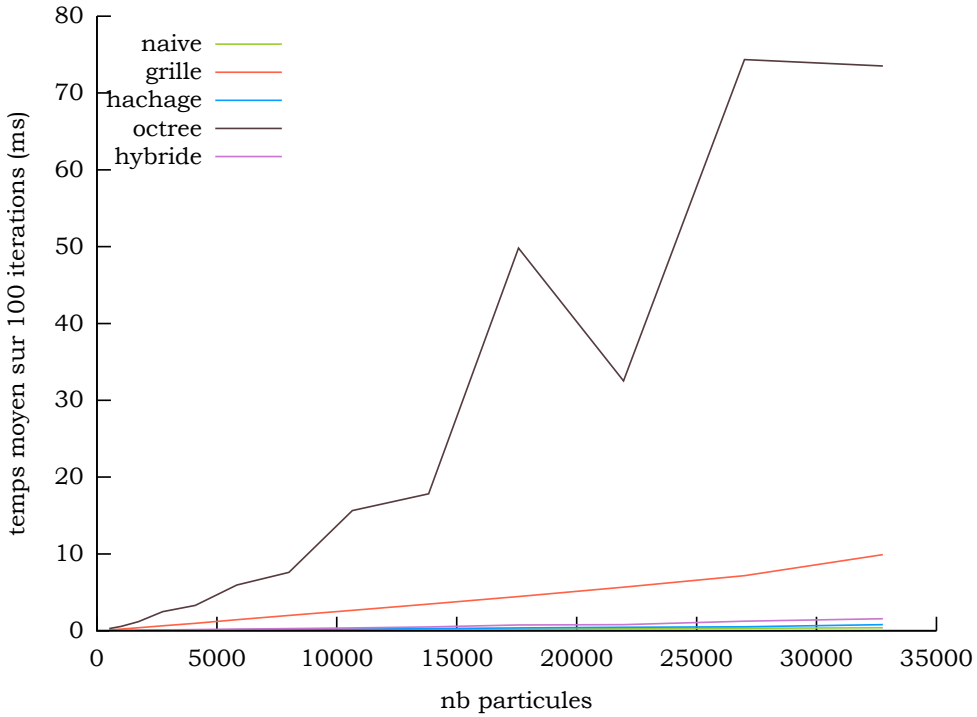
(e) kd-tree

(f) Structure hybride.

FIG. 1.20: Comparaison des structures de recherche pour la perturbation aléatoire et locale.



(a) Comparaison pour simulation des équations de Navier-Stokes.



(b) Comparaison pour perturbation locale.

FIG. 1.21: Comparaison détaillée des structures.

1.5 Choix d'un intégrateur

La méthode *SPH* sert à l'approximation dans l'espace. Il reste donc à traiter les dérivées temporelles afin de pouvoir simuler des matériaux dans le temps.

Nous redonnons les principes de construction des principales méthodes utilisées dans des contextes de simulation temps réel. Ces méthodes sont de type différences finies à une dimension puisque le temps est généralement représenté par une demi-droite discrétisée. Nous conclurons par l'approche retenue.

Selon le principe fondamental de la dynamique, le déplacement d'un corps de masse m est régi par l'équation

$$m\ddot{\mathrm{x}} = \textstyle\sum_i \mathrm{F}_i,$$

où $\mathrm{F_i}$ désigne les forces extérieures exercées sur l'objet, et $\ddot{\mathrm{x}}(t) = \dot{\mathrm{v}}(t) = \frac{d\mathrm{v}(t)}{dt}$ désigne l'accélération du centre d'inertie de l'objet. L'équation différentielle du second ordre précédente est souvent réécrite sous la forme d'un système d'équations différentielles du premier ordre

$$\left\{ \begin{array}{lcl} \dot{\mathrm{x}}(t) & = & \mathrm{v}(t) \\ \dot{\mathrm{v}}(t) & = & \sum_i \mathrm{F}_i/m = f(t, \mathrm{x}(t), \mathrm{v}(t)). \end{array} \right.$$

En utilisant un développement de Taylor au premier ordre, on peut par exemple écrire

$$\mathrm{v}(t+h) \approx \mathrm{v}(t) + h\,\dot{\mathrm{v}}(t),$$

d'où

$$\mathrm{v}(t+h) \approx \mathrm{v}(t) + hf(t, \mathrm{x}(t), \mathrm{v}(t)),$$

qui est la méthode d'Euler explicite (*forward Euler*). On obtient alors aisément

$$\mathrm{x}(t+h) \approx \mathrm{x}(t) + h\mathrm{v}(t+h) \approx \mathrm{x}(t) + h\mathrm{v}(t) + h^2 f(t, \mathrm{x}(t), \mathrm{v}(t)).$$

Le problème de ce type d'approche vient du fait que l'intégration se fait à une constante près (fixée par les conditions initiales) et cette constante est extrapolée en aveugle d'un pas de temps à l'autre [NMK+06].

On peut alors également écrire

$$\mathrm{v}(t) \approx \mathrm{v}(t-h) + h\,\dot{\mathrm{v}}(t),$$

qui donne alors

$$\mathrm{v}(t+h) \approx \mathrm{v}(t) + hf(t+h, \mathrm{x}(t+h), \mathrm{v}(t+h)),$$

appelée méthode d'Euler implicite (*backward Euler*) puisque l'obtention d'une valeur à un pas de temps donné fait intervenir cette même valeur et nécessite ainsi la résolution d'une équation de point fixe.

En traitant les équations dans l'ordre inverse,

$$\left\{ \begin{array}{lcl} \dot{\mathrm{v}}(t) & = & f(t, \mathrm{x}(t), \mathrm{v}(t)) \\ \dot{\mathrm{x}}(t) & = & \mathrm{v}(t), \end{array} \right.$$

et en utilisant l'écriture des différences finies au pas précédent, on voit qu'on peut utiliser une formulation explicite pour le calcul de la vitesse et une formulation implicite la mise à jour des positions, appelée méthode d'Euler semi-implicite (*forward-backward Euler*) *i.e.*

$$\left\{ \begin{array}{rcl} \mathrm{v}(t+h) & = & \mathrm{v}(t) + hf(t, \mathrm{x}(t), \mathrm{v}(t)) \\ \mathrm{x}(t+h) & = & \mathrm{x}(t) + h\mathrm{v}(t+h). \end{array} \right.$$

qui a l'avantage de ne pas nécessiter de résolution d'équation.

Les méthodes précédentes ne font intervenir qu'une approximation de Taylor au premier ordre. Il est également possible d'utiliser une approximation d'ordre supérieur pour obtenir de meilleurs résultats. Par exemple, en utilisant un développement limité à l'ordre 3, on obtient :

$$\begin{array}{rcl} \mathrm{x}(t+h) & \approx & \mathrm{x}(t) + h\mathrm{v}(t) + \frac{h^2}{2}\dot{\mathrm{v}}(t) + \frac{h^3}{6}\ddot{\mathrm{v}}(t) \\ \mathrm{x}(t-h) & \approx & \mathrm{x}(t) - h\mathrm{v}(t) + \frac{h^2}{2}\dot{\mathrm{v}}(t) - \frac{h^3}{6}\ddot{\mathrm{v}}(t) \end{array}$$

soit, en additionnant membre à membre,

$$\mathrm{x}(t+h) \approx 2\mathrm{x}(t) - \mathrm{x}(t-h) + h^2 f(t, \mathrm{x}(t), \mathrm{v}(t)).$$

Si les forces extérieures ne dépendent pas de la vitesse, *i.e.* $f(t, \mathrm{x}(t), \mathrm{v}(t)) = f(t, \mathrm{x}(t))$, cette formulation, dite de Verlet [Ver67], devient entièrement indépendante de la vitesse. La vitesse est gérée implicitement. Ceci présente un avantage important lorsque le système doit respecter des contraintes puisqu'on peut alors simplement modifier la position sans se soucier outre mesure de la vitesse [Jak03].

Il est aussi possible de désynchroniser les évaluations de position et de vitesse. En considérant des demis pas de temps dans la formulation de Verlet, il vient

$$\begin{array}{rcl} \mathrm{x}(t+\frac{h}{2}) & \approx & \mathrm{x}(t) + \frac{h}{2}\mathrm{v}(t) + \frac{h^2}{8}\dot{\mathrm{v}}(t) \\ \mathrm{x}(t-\frac{h}{2}) & \approx & \mathrm{x}(t) - \frac{h}{2}\mathrm{v}(t) + \frac{h^2}{8}\dot{\mathrm{v}}(t), \end{array}$$

soit, en soustrayant membre à membre et en décalant les indices de $\frac{h}{2}$

$$\mathrm{x}(t+h) = \mathrm{x}(t) + h\mathrm{v}(t + \frac{h}{2}).$$

On peut également écrire

$$\mathrm{v}(t + \frac{h}{2}) \approx \mathrm{v}(t - \frac{h}{2}) + hf(t, \mathrm{x}(t), \mathrm{v}(t - \frac{h}{2})).$$

Ce schéma est appelé « saute-mouton » (*leapfrog*) [CvG07] du fait de l'entrelacement entre les évaluations de vitesse et de position

$$\begin{array}{rcl} \mathrm{v}(t+\frac{h}{2}) & \approx & \mathrm{v}(t-\frac{h}{2}) + hf(t, \mathrm{x}(t), \mathrm{v}(t-\frac{h}{2})) \\ \mathrm{x}(t+h) & \approx & \mathrm{x}(t) + h\mathrm{v}(t+\frac{h}{2}). \end{array}$$

D'autres méthodes, comme Runge-Kutta ou des méthodes à pas multiples, permettant une plus grande précision localement, pourraient être envisagées. Néanmoins ces méthodes nécessitent également un nombre d'évaluations de fonction plus important, voire incompatible avec le temps réel. Nous avons donc d'emblée écarté

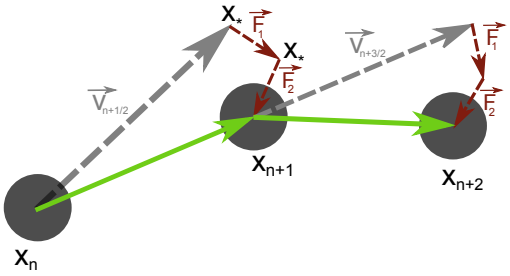


FIG. 1.22: Schéma d'intégration par prédiction-correction.

ces méthodes.

Les formulations présentées sont toutes relativement proches mais présentent des propriétés très différentes. Stern et Desbrun [SD06] ont montré que la méthode d'Euler-explicite explose, la méthode d'Euler-implicite dissipe l'énergie et la méthode d'Euler semi-implicite est symplectique et conserve exactement l'énergie. Les formulations de Verlet présentées sont également symplectiques. Cette propriété semble généralement intéressante. Néanmoins, Stam [Sta99] note que la dissipation numérique peut être souhaitable dans un contexte de simulation temps réel de fluides. Il propose alors un schéma d'intégration de nature implicite pour son approche de simulation semi-lagrangienne.

Clavet *et al.* [CBP05] reprennent les idées du schéma de Stam et les appliquent aux simulations purement lagrangiennes. Il s'agit d'utiliser une approche de type « saute-mouton » mais, pour chaque force appliquée, les particules sont directement déplacées. Cette approche est équivalente à une approche implicite dans laquelle les forces autres que la première seraient considérées comme des contraintes à satisfaire. De cette façon chaque force prend en compte les déplacements opérés par les autres forces (fig. 1.22). Les fonctions mises en jeu sont définies sur tout l'espace de simulation, le schéma est donc valide.

Le schéma proposé nécessite un stockage de positions temporaires afin de ne pas biaiser les calculs par un ordre arbitraire de traitement et de déplacement des particules et nécessite, selon le nombre de forces appliquées, plus de recherches de voisinages. Néanmoins, ceci n'est plus gênant avec une implémentation sur *GPU* où il est plus efficace de répéter des calculs « simples » plutôt que de stocker des résultats intermédiaires, qui, dans notre cas, sont dynamiques. Sur une architecture *GPU*, il est ainsi plus simple de calculer le voisinage de particules pour chaque force à appliquer plutôt que de stocker le voisinage de chaque particule pour un pas de temps. Nous avons donc retenu ce schéma d'intégration de type prédiction-correction.

On remarque que dans le cas où une seule force est appliquée au système, le schéma est équivalent au schéma « saute-mouton » et est donc symplectique.

1.6 Conclusion

Ce chapitre nous a permis de définir les bases de notre modélisation. Nous avons montré l'intérêt de l'utilisation d'une approche lagrangienne dans notre contexte de création de formes libres. Nous avons ensuite présenté la méthode *SPH*, permettant l'approximation de fonctions spatiales ou de grandeurs différentielles de fonctions

Algorithme 1 Schéma d'intégration numérique type « prédiction-correction » .

$$
\begin{aligned}
&\mathrm{x}_* = \mathrm{x}_n + v_{n-1/2}\Delta t \\
&\mathrm{x}_* \leftarrow \mathrm{x}_* + \Delta t^2\, \mathbf{F}^1(\mathrm{x}_*)/m \\
&\mathrm{x}_* \leftarrow \mathrm{x}_* + \Delta t^2\, \mathbf{F}^2(\mathrm{x}_*)/m \\
&\mathrm{x}_* \leftarrow \mathrm{x}_* + \Delta t^2\, \mathbf{F}^3(\mathrm{x}_*)/m \\
&\ldots \\
&\mathrm{x}_{n+1} = \mathrm{x}_* \\
&v_{n+1/2} = (\mathrm{x}_{n+1} - \mathrm{x}_n)/\Delta t
\end{aligned}
$$

spatiales. La méthode repose principalement sur un choix de noyau pour lequel nous avons démontré l'intérêt du noyau *spiky* dans notre contexte.

Les calculs effectués par la méthode *SPH* nécessitent, pour être efficaces, la capacité de rechercher le voisinage spatial d'un point. Nous avons donc présenté les principales structures spatiales répondant à cette problématique et avons défini des stratégies efficaces de recherche dans des structures d'arbre. Les tests que nous avons effectués ont cependant montré que la structure de table de hachage est, en moyenne, la plus efficace.

Dans un souci de généralité et afin de pouvoir tourner sur des machines quelconques, nous avons considéré du matériel informatique standard et nous sommes intéressés uniquement à une implémentation sur le processeur central, *Central Processing Unit* (*CPU*).

Enfin, pour terminer l'ensemble des outils nécessaires pour la simulation de phénomènes physiques, nous avons présenté un schéma d'intégration, de type prédiction-correction.

Nous allons, à présent, voir comment utiliser les outils définis dans ce chapitre, pour simuler effectivement des comportements physiques de matériaux sur des systèmes de particules.

Chapitre 2

Simulation de matériaux par systèmes de particules

Le chapitre précédent nous a fourni des outils permettant de manipuler des équations en temps et en espace en approche lagrangienne.

Nous souhaitons reproduire des interactions naturelles dans un contexte de création de formes tridimensionnelles libres. Dans le monde réel, des matériaux de type pâte à modeler sont usuellement utilisés pour cette tâche. Nous allons ainsi mettre en oeuvre les outils présentés avec comme objectif de simuler des comportements de type pâte à modeler.

Nous étendrons ensuite ces comportements en donnant un cadre, dans lequel il est possible de synthétiser un matériau ayant des propriétés désirées, par assemblage de comportements élémentaires.

2.1 Vers un premier modèle de pâte à modeler

À notre connaissance, il n'existe pas de modèle physique précis, décrivant le comportement des matériaux de type pâte à modeler. Néanmoins, nous avons vu en introduction quelques uns des comportements qui rendent ce matériau attrayant pour la création de formes :

- *hautement déformable* : la pâte à modeler peut subir n'importe quelle déformation, changer de topologie et prendre n'importe quelle forme ;
- *plastique* : les déformations appliquées au matériau sont conservées ;
- *incompressible* : la manipulation du matériau conserve le volume de matériau.

En l'absence de modèle exact, nous pouvons tenter de modéliser la pâte à modeler par l'adjonction d'un modèle répondant à chacune des caractéristiques précédentes.

Tentons, dans un premier temps, de répondre à la caractéristique de déformabilité. Les fluides peuvent être vus comme des matériaux très déformables. Par exemple, les molécules d'eau n'ont pas de liaison forte ce qui permet au fluide de prendre la forme de n'importe quel contenant. Nous allons donc commencer par nous intéresser à la modélisation des écoulements fluides.

2.1.1 Simulation de fluide

Nous souhaitons utiliser les équations des écoulements fluides pour modéliser un matériau hautement déformable. La mécanique des fluides est un domaine vaste et complexe. Nous renvoyons à Batchelor [Bat67] pour une introduction au sujet. Nous exploitons directement les résultats connus du domaine.

L'écoulement fluide est modélisé par les équations de Navier-Stokes, qui, pour un fluide incompressible s'écrivent :

$$\begin{cases} \rho \dfrac{D\mathrm{v}}{Dt} &= -\nabla P + \nu \Delta \mathrm{v} + F \\ \nabla \cdot \mathrm{v} &= 0, \end{cases} \tag{2.1}$$

où v représente la vitesse, P représente la pression, ρ la densité, ν la viscosité, et F les forces extérieures.

La première équation exprime la loi de conversation des quantités de mouvement. Dans un formalisme lagrangien, nous avons vu dans la section 1.2 que la dérivée matérielle du champ de vitesse s'obtient simplement par :

$$\frac{D\mathrm{v}}{Dt} = \frac{\partial \mathrm{v}}{\partial t}.$$

La seconde est appelée équation d'incompressibilité. Elle exprime la conservation de volume au cours du temps. Le volume est directement lié à la masse par la relation

$$m = \rho V.$$

Dans le formalisme lagrangien, les masses sont portées par les particules. Le volume est alors conservé dès lors que le nombre de particules reste constant. Nous pouvons donc, en première approximation, considérer que l'équation d'incompressibilité est gérée implicitement.

Par ailleurs, la viscosité n'est pas un comportement physique essentiel de la pâte à modeler. On considère ainsi des fluides parfaits ce qui revient à fixer $\nu = 0$. Enfin, on peut considérer un milieu idéal dans lequel aucune force externe ne s'applique.

En prenant en compte ces remarques, nous pouvons réécrire les équations de Navier-Stokes :

$$\rho \frac{\partial \mathrm{v}}{\partial t} = -\nabla P. \tag{2.2}$$

On obtient finalement l'équation des écoulements fluides d'Euler.

La méthode *SPH* nous fournit un moyen d'évaluer la densité

$$\rho_i = \sum_{j \in \mathcal{N}(i)} m_j W(\mathrm{r}_i - \mathrm{r}_j, h),$$

où le noyau utilisé est le noyau *spiky*

$$W(\mathrm{r}, h) = \begin{cases} \frac{15}{2\pi h^3} \left(1 - \frac{||\mathrm{r}||}{h}\right)^2 & \text{si } ||\mathrm{r}|| < h \\ 0 & \text{sinon.} \end{cases}$$

La méthode permet également d'évaluer le gradient d'une fonction évaluable dans l'espace. Par conséquent, en modélisant la pression P en tout point du fluide,

la méthode *SPH* nous permettra d'évaluer le gradient ∇P et nous serons en mesure de simuler l'équation d'Euler.

Une première formule proposée pour cette équation d'état est l'équation de Tait [Bat67, BT07])

$$P = P_0 \left[\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right] \tag{2.3}$$

avec, classiquement dans la littérature, $\gamma = 7$. Le calcul de la pression doit être réalisé une fois par particule et par pas de temps. Il faut donc que le calcul soit le plus simple et le plus rapide possible. La formule proposée ici convient mal puisqu'elle nécessite une division et une élévation à une grande puissance, calculs coûteux et sources d'erreur en arithmétique flottante.

Desbrun [Des97] propose une formule plus simple et numériquement plus stable, inspirée de la physique des gaz parfaits :

$$P = k(\rho - \rho_0). \tag{2.4}$$

Harada *et al.* [HKK07] proposent une formulation affine

$$P = P_0 + k(\rho - \rho_0),$$

où P_0 représente la pression du fluide au repos.

Concrètement, le système va chercher à rester proche de la densité d'équilibre ρ_0. Si la densité est supérieure ($\rho > \rho_0$), la pression engendrée est positive et donc on a une force répulsive. Inversement si la densité est inférieure ($\rho < \rho_0$) alors la pression est négative et donc il y a attraction des particules avoisinantes pour tenter de se ramener à une densité d'équilibre.

Notons que l'équation de Tait est, dans sa forme, similaire au potentiel de Lennard-Jones, tandis que l'équation des gaz parfaits s'apparente à la loi de Hooke.

Le calcul du gradient de pression est finalement obtenu par

$$\nabla P_i = \sum_j \frac{m_j}{\rho_j} P_j \nabla W(\mathrm{r}_i - \mathrm{r}_j, h),$$

où

$$\nabla W(\mathrm{r}, h) = \begin{cases} -\frac{15}{\pi h^4} \left(1 - \frac{||\mathrm{r}||}{h}\right) \frac{\mathrm{r}}{||\mathrm{r}||} & \text{si } ||\mathrm{r}|| < h \\ 0 & \text{sinon.} \end{cases}$$

La formule obtenue n'est pas symétrique puisqu'il n'y a aucune raison d'avoir $\rho_i = \rho_j$. Ceci peut engendrer la non conservation des moments. La seconde règle d'or (sous-section 1.3.1) permet de corriger le problème. En écrivant

$$\frac{\nabla P}{\rho} = \nabla \left(\frac{P}{\rho} \right) + \frac{P \nabla \rho}{\rho^2},$$

on obtient la formule d'interpolation pour la particule i :

$$\frac{\partial v_i}{\partial t} = -\frac{\nabla P_i}{\rho_i} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W(\mathrm{r}_i - \mathrm{r}_j, h). \tag{2.5}$$

Par conséquent les forces engendrées par la pression seront de la forme

$$m_i \frac{\partial v}{\partial t} = -m_i \frac{\nabla P_i}{\rho_i} = -\sum_j m_i m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \nabla W(\mathrm{r}_i - \mathrm{r}_j, h),$$

et sont donc bien symétriques.

Nous traitons la formule en deux fois. Chacune des deux contributions,

$$m_i m_j \frac{P_i}{\rho_i^2} \nabla W(\mathrm{r}_i - \mathrm{r}_j, h) \quad \text{et} \quad m_i m_j \frac{P_j}{\rho_j^2} \nabla W(\mathrm{r}_i - \mathrm{r}_j, h),$$

est calculée lors du traitement de la particule correspondante.

Nous obtenons ainsi la simulation d'un fluide parfait. Le système de particules tend naturellement à prendre la forme d'une boule en l'absence de gravité. Toutefois, les déformations subies par un fluide sont temporaires. Une fois le « déformant » enlevé, le fluide reprend une forme sphérique. Nous allons donc voir à ajouter des comportements plastiques pour conserver de façon permanente les déformations appliquées.

2.1.2 Modèle simplifié de plasticité

La plasticité se définit comme la capacité d'un matériau à conserver les déformations qui lui sont appliquées. On renvoie au cours d'Oudin [Oud10] pour une introduction détaillée du comportement plastique. Cette propriété est donc l'opposé de l'élasticité qui est la capacité d'un matériau à reprendre son état initial après déformation.

La modélisation de l'élasticité remonte à Hooke et au célèbre :

> *Ut tensio, sic vis.*
>
> Telle extension, telle force.

La loi de Hooke est généralement illustrée par l'exemple d'un ressort subissant une élongation :

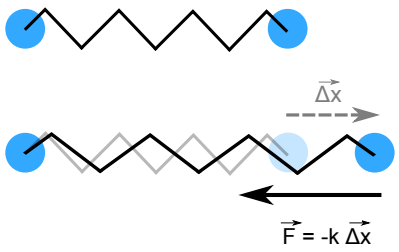


FIG. 2.1: Modélisation de l'élasticité par ressort (loi de Hooke unidimensionnelle).

Plus généralement, la loi établit la relation linéaire entre la déformation et la contrainte subie par un matériau élastique, soit

$$\sigma = E\varepsilon, \tag{2.6}$$

où E est un tenseur d'ordre 4, qui, dans le cas d'un matériau isotrope comme la pâte à modeler, dépend de deux paramètres :

- module de Young, mesurant la tendance à l'allongement ;
- coefficient de Poisson, exprimant la tendance au rétrécissement consécutive à un allongement.

L'utilisation de la loi de Hooke généralisée à $\mathbb{R}^3$ a été étudiée dans le cadre de simulations réalistes sur des systèmes de particules [Kei06, Bec09, Len09]. Les calculs mis en jeu sont alors coûteux et peu appropriés pour le temps réel.

Nous avons préféré opter pour une autre modélisation, plus simple, pour répondre à la contrainte de simulation temps réel. Nous considérons que chaque particule représente un volume élémentaire de matière. Les éléments peuvent ensuite s'agencer en réseaux unidimensionnels, bidimensionnels ou tridimensionnels. Nous considérons, à l'instar de Martin *et al.* [MKB+10] qu'un réseau de dimension n est une agrégation de réseaux de dimension $n-1$. Nous utilisons alors une modélisation unidimensionnelle pour l'élasticité, équivalente à la loi des ressorts (figure 2.1).

Nous allons adapter la loi de Hooke pour simuler la plasticité. O'Brien *et al.* [OBH02] décrivent le phénomène de plasticité selon le critère de Von Mises. Un matériau soumis à déformation subit, dans un premier temps, une déformation élastique réversible. Une fois le seuil d'élasticité dépassé, si le matériau est :

fragile, il y a fracture ;

ductile, le matériau commence par être en phase plastique, conservant les déformations. Ces déformations plastiques modifient généralement le seuil d'élasticité (on parle alors d'écrouissage). Les déformations plastiques sont conservées jusqu'à atteindre le seuil de rupture.

La plasticité fait donc intervenir deux coefficients : le seuil de plasticité γ_p, et le seuil de rupture γ_r. On voit ainsi, qu'en comparant l'allongement subit par le ressort à ces deux coefficients, on peut modéliser les phénomènes plastiques au moyen de ressort de longueur variable (fig. 2.2) :

- si le déplacement relatif de particules reliées par un ressort est inférieur à γ_p, les particules sont déplacées selon la loi de Hooke ;
- lorsque le déplacement relatif est supérieur à γ_p et inférieur à γ_r, la longueur à vide du ressort est réinitialisée à la distance courante entre les particules ;
- lorsque le déplacement relatif dépasse γ_r, le ressort est supprimé.

Cette modélisation, étendant celle de Clavet *et al.* [CBP05], permet ainsi de modéliser :

- des matériaux purement plastiques en prenant $\gamma_p = 0$;
- des matériaux fragiles en prenant $\gamma_r = \gamma_p$;
- des matériaux ductiles et la rupture en prenant $\gamma_p < \gamma_r$.

Pour gérer l'adjonction de morceaux de matière initialement séparés, des ressorts sont ajoutés localement, en fonction du voisinage.

La combinaison de la simulation de fluide et de la plasticité donne des résultats convenables. La pâte à modeler étant presque purement plastique, nous prenons $\gamma_p \approx 0$. On constate cependant que le volume n'est pas toujours conservé. Le gradient de pression empêche les accumulation globales de matière. L'interpénétration des volumes associés aux particules n'est pas explicitement requise ce qui donne lieu à des pertes locales de volume. Nous allons donc ajouter un comportement pour simuler l'incompressibilité.

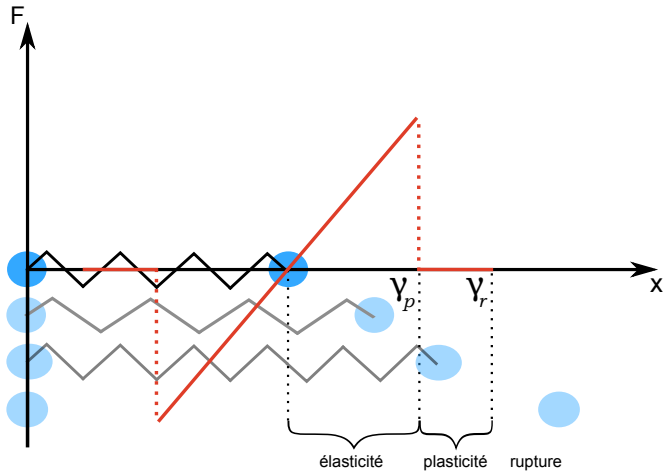


FIG. 2.2: Modèle de plasticité.

2.1.3 Incompressibilité

La conservation du volume de matière, autrement dit l'incompressibilité, est une propriété souvent désirée car elle caractérise fortement les matériaux qui nous entourent. Nous avons vu, lors du traitement des équations de Navier-Stokes, que le formalisme lagrangien permettait de passer outre la considération des équations de conservation de volume dès lors que le nombre de particules était fixe au cours de la simulation. En fait, chaque particule représente un volume élémentaire de matière et, sans ajouter de contrainte, il est possible que les volumes des particules s'interpénètrent entraînant ainsi une réduction du volume total.

La propriété d'incompressibilité n'est pas explicitement modélisée et par conséquent n'est pas toujours respectée. La quantité de matière est préservée mais le volume global peut être affecté. L'utilisation d'un noyau *spiky* pour empêcher l'annulation des forces de pression dans le cas de particules très proches ne suffit pas à lever le problème.

Becker et Teschner [BT07] soulignent l'importance de l'équation d'état utilisée pour calculer les forces de pression. L'utilisation de l'équation de Tait, très raide, leur permet d'obtenir des fluides peu compressibles.

Solenthaler et Pajarola [SP09] déduisent à partir des variations de densité et avec un schéma de prédiction-correction, les forces de pression à appliquer.

Dans notre contexte, nous ne souhaitons pas effectuer une simulation exacte mais avoir un comportement proche du matériau réel. Clavet *et al.* [CBP05] proposent d'ajouter une force virtuelle. L'idée consiste à repousser les particules trop proches les unes des autres ; on introduit l'équivalent d'une force de pression interne purement répulsive, qualifiée de pression de proximité.

La densité de proximité doit être calculée avec un noyau décroissant très rapidement vers 0 pour que seules les particules proches aient un poids important

$$W_{\text{near}}(r,h) = \frac{15}{\pi h^3}\left(1 - \frac{||r||}{h}\right)^3. \tag{2.7}$$

La pression de proximité est ensuite obtenue par

$$P_{\text{near}} = k_{\text{near}}\rho_{\text{near}} \tag{2.8}$$

où ρ_{near} est la densité de proximité calculée via le noyau d'interpolation W_{near} et k_{near} est un coefficient. Les forces générées seront ainsi purement répulsives et imiteront l'incompressibilité des matériaux.

L'utilisation des comportements de fluide parfait, plasticité et d'incompressibilité fournit un matériau aux propriétés attendues :

- le matériau peut subir toute sorte de déformations, y compris des changements de topologie ;
- les déformations appliquées sont conservées en l'absence de nouvelles déformations ;
- le volume est globalement conservé au cours des déformations.

Nous répondons ainsi à notre objectif premier de simulation de pâte à modeler virtuelle. Nous allons voir que cette solution d'assemblage de propriétés peut être étendue pour définir de nouveaux comportements.

2.2 Modèle de matériau

Notre objectif premier était de modéliser un matériau ayant des propriétés semblables à la pâte à modeler. Il semble néanmoins intéressant de réfléchir à un cadre plus général de comportement de matériaux.

De notre premier modèle, on peut retenir principalement deux éléments :

1. la déformabilité est un comportement global du matériau ; elle implique que chaque élément de matière peut se retrouver au voisinage de n'importe quel autre élément.
2. la plasticité régit les comportements de façon plus locale ; elle est modélisée par un ressort entre des particules voisines.

Nous avons donc un modèle faisant apparaître deux couches :

- une couche *macroscopique*, définie sur l'ensemble du système, décrivant la topologie du système (au sens du voisinage de particules) ;
- une couche *microscopique*, définie par paire de particules, décrivant la géométrie du système par la description des réactions aux forces.

Nous allons voir comment enrichir les couches du modèle et permettre la création d'autres types de matériaux.

2.2.1 Vue macroscopique

La vue macroscopique définit la topologie globale du système de particules, et donc de la forme modélisée. Nous allons voir les différents types de topologie envisageables.

2.2.1.1 Modèle déformable

L'argile ou la pâte à modeler sont clairement des matériaux déformables pouvant représenter des formes de topologie quelconque. Nous avons vu que la déformabilité pouvait être assimilée à des fluides puisque les molécules de ces matériaux sont relativement libres.

Cette liberté topologique est modélisée par des fluides incompressibles non visqueux. Les équations mises en jeu pour ces matériaux sont ainsi les équations d'Euler décrites précédemment

$$\frac{\partial \mathrm{v}}{\partial t} = -\frac{1}{\rho}\nabla P,$$

où P représente la pression interne du fluide.

Nous avons vu que l'incompressibilité pouvait être modélisée par des forces purement répulsives, de courte portée (section 2.1.3).

La topologie, au sens du système de particules, s'adapte aux modifications et permet à la forme d'adopter une topologie quelconque pour la forme modélisée. Les matériaux déformables seront donc les matériaux privilégiés pour la sculpture virtuelle puisqu'ils n'imposent aucune contrainte et permettront la création de formes entièrement libres.

2.2.1.2 Modèle solide

A l'opposé de la topologie libre, il est possible de vouloir figer la topologie. Ceci correspond assez naturellement à des matériaux solides pour lesquelles les molécules sont fortement liées entre elles et dont la structure est fixe. Aucune liaison ne peut être créée spontanément entre des particules voisines.

Par conséquent, dans un matériau solide, les particules auront des liaisons par défaut. On peut imaginer une structure semblable à celle des cristaux avec des particules arrangées en réseau ou, plus simplement, pour chaque particule, créer des liaisons avec l'ensemble de son voisinage en début de simulation (fig. 2.3). De nouvelles liaisons ne peuvent être créées par déformation. Par contre, elles peuvent se défaire si les déformations appliquées sont trop importantes, simulant ainsi des phénomènes de fracture.

L'avantage est qu'il devient très simple de passer d'un matériau déformable à un matériau solide et vice versa. Il suffit, pour solidifier un matériau déformable, de contraindre son voisinage aux particules voisines courantes et de ne plus effectuer la simulation de fluide. Inversement, une particule solide devenant déformable reconstruit son voisinage à chaque pas de temps et est déplacée selon les équations d'Euler.

2.2.1.3 Autres modèles

Les modèles déformables et solides présentent des volumes cohérents. On peut également réfléchir à l'extension des modèles de matériaux à :

- des matériaux non cohérents,
- des matériaux non volumique

de façon à avoir un cadre de travail le plus générique possible.

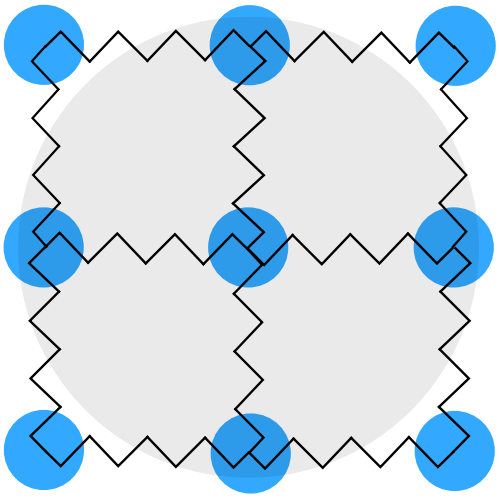


FIG. 2.3: Exemple de topologie de solide.

2.2.1.3.1 Simulation de poudres : Une poudre est un état fractionné de la matière, sous forme de petits morceaux constituant un solide. On peut donc voir une poudre comme un matériau sans topologie. Chaque particule est indépendante des autres. La difficulté de simulation des poudres vient de leur comportement ambivalent : elles peuvent se comporter soit comme des corps rigides (chute des grains de sable dans un sablier), soit comme un ensemble continu (inclinaison du sablier provoquant un écoulement de sable).

Des modélisations ont été proposées pour chacun de ces comportements. Zhu et Bridson [ZB05] proposent une simulation basée sur *SPH* pouvant donc répondre à l'aspect continu et cohérent d'une poudre.

La simulation de poudres de type « corps rigides » nécessite un autre traitement. Cet aspect peut être géré par des méthodes de type *Molecular Dynamics* (MD). Ces méthodes reposent sur la modélisation des contacts entre particules. Deux particules doivent au pire être en contact tangentiel. Des interpénétrations sont possibles temporairement et engendrent des forces normales répulsives et des forces tangentielles pour prendre en compte les phénomènes de friction. Nous renvoyons à [BYM05, LH93] pour plus de détails.

À noter que les méthodes de type *Discrete Element Method* (*DEM*) reposent sur les mêmes fondements en introduisant l'orientation des particules, une gestion plus fine des relations de contact ainsi que des géométries de particules complexes.

Une synthèse de ces modélisations partielles de matériaux granulaires, tous deux particulaires, est envisageable. En effet, d'après Zhu et Bridson [ZB05], il est possible d'utiliser la loi de Mohr-Coulomb pour détecter l'écoulement du matériau granulaire. Si le critère n'est pas rencontré, une poudre peut alors être considérée comme un amas de corps solides et simuler via une *DEM*. Bell *et al.* [BYM05] utilisent des regroupements de quelques particules se déplaçant ensemble comme unité de base de simulation de poudre. Le passage de l'écoulement à la simulation de corps rigides pourraient donc simplement se faire en figeant les liaisons entre des particules voisines de façon similaire au passage d'un matériau déformable à un matériau solide.

2.2.1.3.2 Vers les systèmes de dimensions inférieures : Les poudres permettraient de gérer des systèmes 3D discrets. Cependant, notre modélisation de la plasticité et de l'élasticité reposent sur une vision de la matière par assemblage de « blocs élémentaires » . Par conséquent, il est naturel de vouloir modéliser des systèmes de dimension inférieure. On pourrait ainsi considérer des systèmes à une ou deux dimensions.

Si l'utilisation de poudres, plaques de métal ou d'effets comme la fourrure peuvent être intéressants d'un point de vue artistique, leur représentation non volumique nécessiterait des processus de visualisation dédiés. Aussi, ces extensions n'ont pas été implémentées et constituent des axes de réflexions futures.

2.2.2 Vue microscopique

La vue microscopique régit les liaisons entre paires particules ; c'est cette vue qui définit la réaction du matériau aux sollicitations de l'utilisateur. Elle sera constituée de trois briques de base.

2.2.2.1 Élasticité

> **Élasticité :** Propriété physique d'un corps de reprendre sa forme initiale après suppression de la sollicitation.

Nous utilisons la modélisation linéaire proposée par Hooke pour simuler l'élasticité. La loi de Hooke énonce que l'allongement d'une liaison élastique est proportionnel à la force appliquée. On appelle module de Young et on note E le coefficient de proportionnalité. La loi de Hooke s'écrit

$$E\varepsilon = \sigma, \tag{2.9}$$

avec ε la déformation (caractérisation du changement de forme local en tout point du matériau) et σ la contrainte *i.e.* l'état de sollicitation mécanique — efforts internes — en tout point du matériau.

Nous modélisons l'élasticité par des ressorts unidimensionnels de raideur k. La loi de Hooke s'écrit alors

$$F = -k\,\overrightarrow{\Delta\ell},$$

où $\overrightarrow{\Delta\ell}$ exprime la variation de longueur du ressort par rapport à sa longueur de repos ℓ.

Notre schéma d'intégration consiste à calculer directement, pour chaque force, les déplacements engendrés. Par conséquent, les déplacements élastiques, appliqués à chaque particule, s'écrivent

$$\mathrm{D} = dt^2\, k(\ell - d_{ij})\frac{(\mathrm{x}_j - \mathrm{x}_i)}{d_{ij}}, \tag{2.10}$$

où dt est le pas de temps de simulation.

Cette formulation, très simple, revient à utiliser à une échelle mésoscopique des comportements microscopiques. Le changement d'échelle est généralement modélisé par la généralisation de cette loi et l'utilisation de tenseurs [Bec09]. Néanmoins, ces formulations sont plus complexes et donc coûteuses en temps. De plus, certaines formulations ne sont pas invariantes par transformation rigide du matériau. L'utilisation de ressorts repose uniquement sur une longueur et est donc invariante par translation et rotation.

2.2.2.2 Plasticité

> **Plasticité :** Tendance d'une matière à rester déformée après réduction de la contrainte déformante (contrainte ayant une valeur inférieure ou égale à celle du seuil d'écoulement du matériau).

Le comportement plastique est un comportement indépendant du temps, mais il dépend de l'histoire mécanique du matériau. Contrairement au comportement élastique, il n'y a donc plus de relation biunivoque entre contrainte et déformation. Comme nous l'avons vu, nous modélisons la plasticité par des ressorts déformables à l'instar de Clavet *et al.* [CBP05].

La plasticité ne génère pas directement de déplacement mais elle met à jour les propriétés des ressorts des particules incriminées. La rupture est modélisée en supprimant les ressorts pour des particules trop éloignées.

Nous utilisons finalement un modèle unifié élasto-plastique dont les paramètres permettent de faire varier le comportement de l'élasticité pure à la plasticité pure.

2.2.2.3 Viscosité

> **Viscosité :** Propriété de résistance à l'écoulement uniforme et sans turbulence se produisant dans la masse d'une matière.

La viscosité se manifeste par la sensibilité de la réponse à la vitesse de sollicitation. Les équations de Navier-Stokes pour les fluides incompressibles modélisent la viscosité par

$$\mu \Delta v$$

où μ est un coefficient de viscosité et $\Delta = \nabla \cdot \nabla$ est le laplacien. On pourrait alors directement utiliser *SPH* pour calculer ce terme de viscosité en transférant l'opérateur de laplacien sur le noyau. Néanmoins, Monaghan et Gingold [MG83] ont montré qu'une formulation directe de la viscosité par *SPH* introduit des oscillations non physiques. Ils proposent d'utiliser une force liée à la vitesse relative des particules, et approchent le calcul du gradient de vitesse par

$$\nabla \mathrm{v}_i \approx \frac{\mathrm{v}_j - \mathrm{v}_i}{||\mathrm{x}_j - \mathrm{x}_i||}.$$

Benz [Ben90b] synthétise l'approche :

$$\mu_{ij} = \frac{h(\mathrm{v}_i - \mathrm{v}_j) \cdot (\mathrm{x}_i - \mathrm{x}_j)}{||\mathrm{x}_i - \mathrm{x}_j||^2 + \varepsilon h^2},$$

où le terme ε est utile pour éviter les problèmes numériques dans le cas de particules trop proches. Afin de n'ajouter que de l'entropie au système, la viscosité affecte seulement des particules ayant une dynamique de rapprochement, on introduit donc

$$\Pi_{ij} = \begin{cases} \frac{-\lambda \mu_{ij} + \beta \mu_{ij}^2}{\rho_i + \rho_j} & \text{si} \quad (\mathrm{v}_i - \mathrm{v}_j) \cdot (\mathrm{x}_i - \mathrm{x}_j) < 0, \\ 0 & \text{sinon} \end{cases}$$

et on obtient, dans notre schéma prédictif-correctif, des déplacements radiaux

$$D^{\text{visc}} = dt^2 m_i \sum m_j \Pi_{ij} \nabla W(||\mathrm{x}_i - \mathrm{x}_j||, h), \tag{2.11}$$

dûs à la viscosité et contrôlés par les paramètres λ et β.

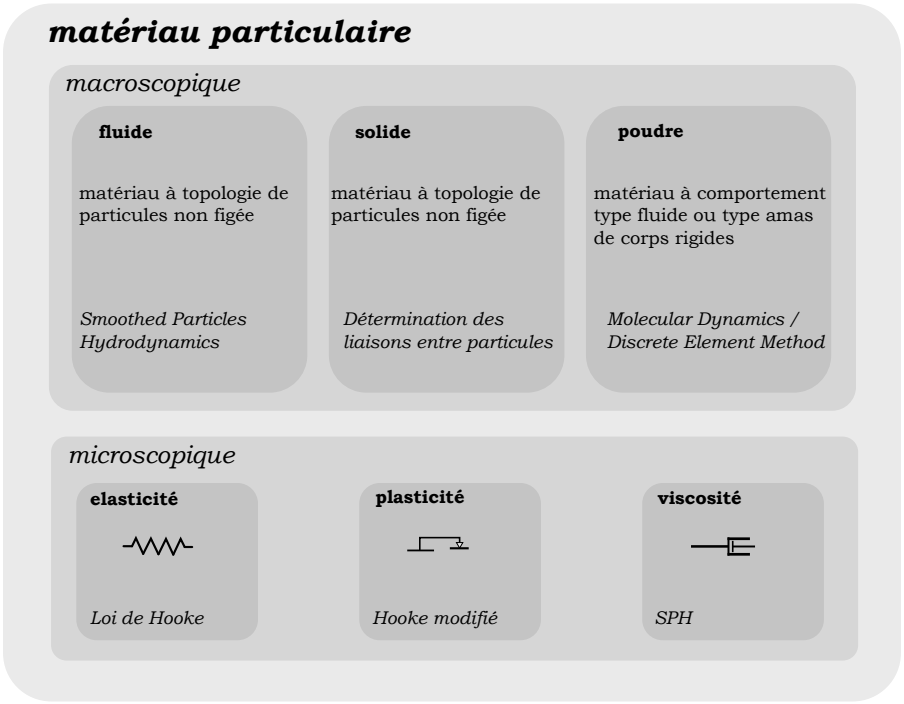


FIG. 2.4: Modèle de matériau particulaire à deux couches.

2.3 Conclusion

Nous avons présenté un modèle de matériau pour la simulation de pâte à modeler. Notre modélisation de la pâte à modeler passe par la combinaison des trois comportements fondamentaux, déformabilité, plasticité et incompressibilité. L'observation de cette modélisation fait ressortir un modèle à deux couches : l'une, qualifiée de macroscopique, gérant la topologie globale du système de particules, et l'autre, microscopique, régissant plus localement sur la géométrie des particules du système. Ce modèle de base fait intervenir les équations d'Euler pour modéliser l'aspect déformable du matériau et un modèle simple de plasticité pour conserver les déformations.

Nous avons ensuite cherché à étendre ce modèle afin de disposer d'une palette riche de comportements de matériaux (fig. 2.4). L'élasticité et la plasticité sont gérées par des ressorts et la viscosité est interprétée dans le formalisme *SPH*. Notre modèle est ainsi plus simple que les modèles génériques qui ont pu émerger récemment comme ceux de Lenaerts [Len09], Keiser [Kei06] ou Becker [Bec09]. Il ne repose que sur des interactions entre paire de particules là où des modèles plus complexes reposent sur des évaluations tensorielles. En plus de la formulation très simple, donc appropriée au temps réel, cette modélisation est indépendante des transformations rigides subies par le matériau, contrairement à certaines formulations tensorielles.

Les comportements physiques observés lors d'une simulation de notre modèle sont conformes à ceux attendus lors de l'utilisation de pâte à modeler. Notre modèle fournit ainsi les résultats physiquement réalistes attendus, et permet une simulation temps réel.

L'utilisation de la physique, en tant que telle, peut être vue comme un outil direct de création de formes artistiques. L'utilisation des paramètres permet de créer

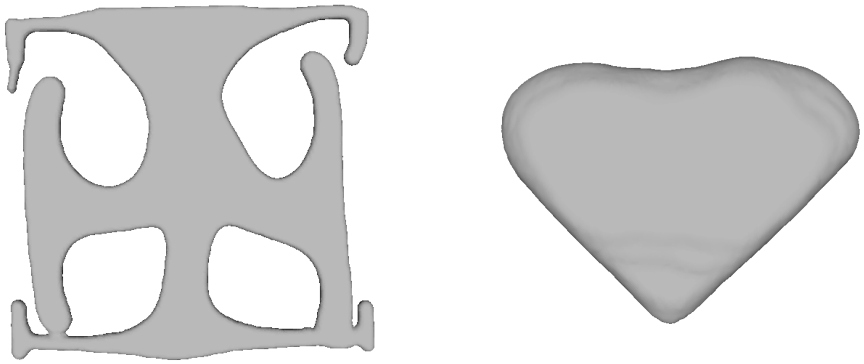


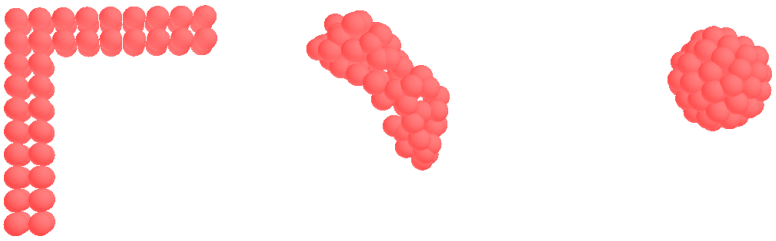
FIG. 2.5: Création de formes par la physique des matériaux par simulation de fluide. À droite, résultat obtenu en augmentant l'incompressibilité.

aisément des formes différentes (fig. 2.5) pour permettre de faire ce que Dönni [Dö09] qualifie de *governed design*.

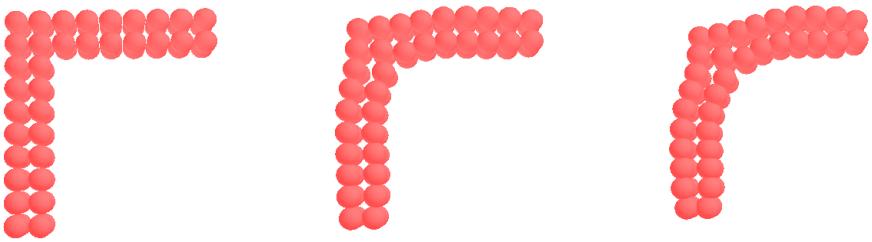
Néanmoins, dans un contexte de *modeling*, certains de nos tests ont montré que la physique pouvait générer des réactions non attendues par l'utilisateur. À la vue de ces difficultés, nous avons finalement préféré considérer la physique non pas comme un mode par défaut mais comme un outil. La simulation de fluide peut aisément être vue comme un outil de lissage de forme (fig. 2.6). Nous avons donc modifié l'approche initiale.

L'approche lagrangienne permet une modélisation intuitive de volume. Par défaut, les particules ne sont plus régies par des lois physiques mais représentent des *descripteurs* de la forme modélisée. La matière modélisée est inerte par défaut. Les descripteurs servent également de *canevas* sur lequel appliqué des comportements physiques, temporaires ou permanents, décrits dans ce chapitre. La physique peut ainsi devenir un outil de modification de forme, utilisable comme un pinceau.

Nous avons, jusqu'à présent, défini un cadre mathématique d'utilisation des systèmes de particules, puis, dans ce cadre, nous avons défini une modélisation de comportements physiques. Ces comportements modélisent ceux de matériaux continus. Les systèmes de particules sont une discrétisation de la matière. Nous allons à présent voir comment reproduire l'aspect continu des matériaux modélisés.



(a) Lissage par comportement fluide.



(b) Lissage par comportement de pâte à modeler.

FIG. 2.6: Lissage de système de particules par comportements physiques.

Chapitre 3

Habillage de système de particules

Jusqu'à présent, nous nous sommes uniquement intéressés à la modélisation d'un matériau par un système de particules. L'utilisation d'un système de particules permet de gérer intrinsèquement des topologies quelconques de forme. La question de la visualisation de notre matériau se pose à présent.

Dans notre contexte, le système pourra évoluer :

- soit sous l'effet de la simulation numérique de matériau ;
- soit sous l'effet de l'utilisateur, qui pourra déformer le système de particules représentant son matériau virtuel pour le modeler à sa convenance.

L'habillage du système de particules doit être actualisé en temps réel pour permettre à l'utilisateur de manipuler la matière sans gêne.

Nous donnons ici quelques définitions utiles par la suite :

Maillage, maillage triangulaire, maillage quadrangulaire : un maillage est la donnée d'un ensemble de sommets, d'arêtes reliant deux sommets et de faces constituées d'aux moins trois arêtes formant un chemin bouclant.

Un maillage dont les faces sont uniquement définies par trois arêtes est appelé maillage triangulaire ou maillage *tri*. Un maillage dont les faces sont définies par quatre arêtes est appelé maillage quadrangulaire ou maillage *quad*.

Un maillage peut donc être vu comme un graphe. Dans la suite, on parlera principalement de maillage triangulaire en omettant parfois le qualificatif triangulaire.

Géométrie d'un maillage : la géométrie d'un maillage désigne l'ensemble des données géométriques définissant la surface maillée *i.e.* l'ensemble des positions des sommets constituant le maillage.

Topologie d'un maillage : la topologie d'un maillage désigne l'ensemble des données topologiques définissant la surface maillée *i.e.* l'ensemble des arêtes et faces constituant le maillage. Ces données permettent de définir une notion de voisinage sur les éléments du maillage et donc une topologie.

3.1 État de l'existant

Visualisation discrète

Chaque particule de notre système de particules peut être vue comme un petit échantillon de matière, que nous avons supposé sphérique. Un premier choix trivial de visualisation d'un système de particules consiste à utiliser une sphère pour chaque particule. Nous avons fait le choix d'utiliser des matériaux homogènes et on peut donc utiliser des sphères de rayon r_p, centrées sur la position p_i de chaque particule i.

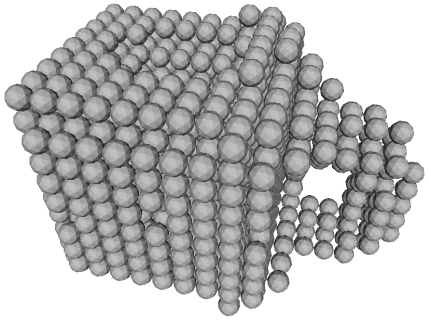


FIG. 3.1: Visualisation d'un système de particules par primitives sphériques.

L'avantage de l'approche est double : la représentation du système est fidèle à la matière réellement modélisée et ne demande aucun coût de calcul. Cependant, cette représentation n'est qu'une discrétisation d'un modèle sous-jacent qui se veut continu et n'est donc pas pleinement satisfaisante comme visualisation de la matière simulée. De plus, la reconnaissance de la forme globale n'est pas toujours simple. Le rendu de la figure 3.1 ne permet pas de reconnaître la partie creuse de la tasse.

Nous allons donc considérer la problématique de visualisation d'un nuage de points et chercher à reconstruire une peau sur le système de particules de façon à rendre compte de l'aspect continu de la matière modélisée.

Comme notre modèle sous-jacent n'est pas structuré, on peut envisager de reconstruire une surface par morceaux. Un choix évident est de chercher à représenter un morceau de surface par particule proche de la surface. La surface la plus simple est le plan et un plan est entièrement déterminé par sa normale et un point appartenant au plan. En estimant une normale aux particules proches de la surface, il devient possible de représenter le système de particules par une collection de plans, appelés *surfel* [PZvBG00] pour les disques modélisés dans la 3D ou *splat* [ZPvBG01] pour le rendu écran final, associés à chaque particule. [KB04] présente une revue des méthodes développées pour la manipulation et le rendu par *splats*.

L'utilisation d'un rendu par *surfels* dans le contexte des systèmes de particules a, entre autres, été abordée dans [Ves04, Din07]. Vesterlund [Ves04] propose d'utiliser des *surfels* situés sur une isosurface représentant la frontière de la fonction caractéristique. Les particules proches de la surface sont d'abord déterminées et génèrent un ensemble initial de *surfels* dont la normale est déterminée par le gradient de la fonction caractéristique. En fonction de la distance évaluée à l'isosurface, chaque *surfel* est divisé en quatre nouveaux *surfels* itérativement. Une mesure de l'isolement des particules est déterminée par étude d'une matrice de covariance. Dans le cas d'une

particule seule, des *surfels* sont générés tout autour de la position. Le rendu des *surfels* est effectué de façon standard.

Dingle [Din07] utilise également des systèmes volumiques de particules mais couplés à des forces de type Lennard-Jones. Dingle détermine les particules de surface en plongeant le système dans une grille et en déterminant l'intersection de la grille avec le système. Les voxels bords contiennent les particules de la surface qui génèrent toutes un *surfel*. Les normales de ces particules sont calculées comme normales du plan tangent au sens des moindres carrés d'un voisinage de particules de surface. La bonne orientation des normales est assurée en utilisant l'information des particules intérieures. Les *surfels* sont ensuite triés en fonction de leur distance au point de vue puis un rendu standard de *splat* est effectué.

Représentation par surface explicite

Boubekeur *et al.* [BRS04] proposent d'effectuer une reconstruction par morceaux non plus basée sur chaque donnée discrète mais sur un regroupement de plusieurs données. Les groupes de données sont déterminés par une voxelisation du nuage de points. La construction des morceaux de surface s'effectue en 4 étapes pour les voxels contenant de l'information :

1. projection des points d'un groupe sur le plan tangent moyen,
2. triangulation de Delaunay dans le plan de l'ensemble des points projetés,
3. reprojection des points en 3D,
4. subdivision de Loop de la triangulation obtenue.

L'utilisation de la projection permet d'éviter une triangulation 3D délicate en considérant que la surface est localement un champ de hauteur (variété de dimension 2). Chaque voxel peut être traité indépendamment des autres. Pour rendre les morceaux de surface de voxels adjacents visuellement jointifs, Boubekeur *et al.* proposent d'utiliser des points contenus dans les voxels voisins.

Destelle *et al.* [DD06] proposent une approche relativement similaire en trois étapes :

1. partitionnement du nuage de points dans une structure spatiale adaptative (type octree),
2. parcours des feuilles non vides de la structure spatiale et construction gloutonne, en parcourant les cellules non vides voisines et sans prendre en compte les données réelles du nuage, d'un premier maillage qui est ensuite simplifié pour constituer un maillage de contrôle,
3. subdivision du maillage simplifié et minimisation de la distance au nuage de points initial.

La surface reconstruite n'est donc pas définie par morceaux.

Sharf *et al.* [SLS+06] proposent également une reconstruction de surface en un morceau en utilisant un front déformable. Le nuage de points est plongé dans une grille. La distance des sommets de la grille à la surface est évaluée en utilisant des fonctions à base radiale à support compact pour déterminer le niveau 0 de la surface puis via propagation par *Fast Marching Method* [Set96, Bær01]. Dans un

premier temps, le front se déplace dans la direction définie par ses normales jusqu'à atteindre approximativement la surface cible. A chaque étape, le front est lissé par moindre carrés. Une fois que tous les points du front sont suffisamment proches de la surface, le maillage est projeté sur les points en entrée pour récupérer les détails de la surface à reconstruire. Pour gérer les topologies quelconques, une gestion de collision de fronts est considérée et permet à deux fronts de se fusionner. La propagation du front dans la grille de distances à la surface peut être vue comme une reconstruction de surface implicite puisque ce ne sont pas la donnée du nuage de points qui est utilisée mais une représentation intermédiaire et discrète.

Étudions de plus près l'intérêt de la représentation par surface implicite.

Représentation par surface implicite

Les approches précédentes tentent d'exprimer explicitement une surface, discrète ou continue, passant par le nuage de points. Une autre approche consiste à utiliser une définition implicite de la surface au moyen de la définition d'un champ scalaire de l'espace

$$\begin{array}{rccc} f & : & \mathbb{R}^3 & \rightarrow & \mathbb{R} \\ & & (x,y,z) & \mapsto & f(x,y,z). \end{array}$$

La surface S, ou isosurface, est alors vue comme l'ensemble des points de l'espace tels que

$$S = \left\{(x,y,z) \in \mathbb{R}^3 \text{ tq } f(x,y,z) = c\right\},$$

où $c \in \mathbb{R}$ est appelé isovaleur. On pourra consulter l'introduction de Bloomenthal sur les surfaces implicites [Blo98].

La reconstruction de surface sur nuage de points par surface implicite s'effectue en deux étapes. Dans un premier temps, on construit une fonction de l'espace permettant de caractériser le nuage de points dont on veut visualiser une représentation surfacique[1] [HDD+92, CBP05, APKG07], puis on extrait une représentation de la surface [LC87, JLSW02].

Il est naturel de vouloir caractériser la distance[2] de tout point de l'espace à la surface recherchée qui devient l'isovaleur 0 du champ construit. Dans le cas d'un nuage de points, comme on dispose de peu d'information, il s'agit de donner une interprétation du nuage par rapport à la surface sous-jacente et d'en déduire une évaluation de la distance au voisinage du nuage.

L'une des premières approches basée sur la reconstruction de surface sur un nuage de points par surface implicite est due à Hoppe *et al.* [HDD+92]. La reconstruction est basée sur l'idée que les points du nuage sont soit sur, soit à proximité de la surface recherchée. Comme la surface à reconstruire est supposée être une variété, une collection d'estimations de plans à la surface est calculée. L'estimation des plans tangents est faîte au sens des moindres carrés. Les points sont regroupés en paquets et la normale d'un paquet est calculée par une analyse de la matrice de covariance des points. Les plans sont ensuite orientés de façon cohérente *i.e.* les normales de plans tangents spatialement proches doivent avoir la même orientation. Pour tout point de l'espace, la distance (signée) évaluée à la surface cible est alors la distance

[1] À noter que les surfaces implicites sont également utilisées pour visualiser des surfaces dont on connaît l'expression formelle ; la construction de la fonction de l'espace consiste alors simplement à évaluer la fonction sur un ensemble discret de points de l'espace.

[2] si la distance est signée, les valeurs négatives représentent alors généralement l'intérieur du volume défini par la surface et les distances positives représentent l'extérieur.

signée au plan tangent le plus proche. Les bords sont détectés en prenant en compte la densité du nuage initial et une évaluation du bruit dans le nuage.

Avec l'émergence des simulations de fluides basées sur des approches lagrangiennes, la littérature a également exploré la reconstruction de surface sur des nuages volumiques de points *i.e.* dont les points ne sont plus supposés proches de la surface recherchée et pouvant avoir des caractéristiques physiques. L'une des premières approches interactives est due à Müller *et al.* [MCG03] et repose sur le concept de champ de couleur, *color field*. Le champ *color field* peut être vu comme une évaluation de la fonction indicatrice du volume soit la fonction valant 1 aux positions des particules et 0 partout ailleurs. Pour évaluer la fonction à l'ensemble de l'espace, l'interpolation *SPH* est utilisée

$$\mathbf{c}(\mathrm{x}) = \sum_{\mathcal{V}(\mathrm{x})} m_i \frac{\mathbf{1}}{\rho_i} W(\mathrm{x} - \mathrm{x}_i). \tag{3.1}$$

Zhu et Bridson [ZB05] proposent une approche relativement proche de celle de Hoppe et cherchent à reconstruire le champ signé de distance à la surface. Ils souhaitent visualiser la surface épousant chaque particule de la frontière. Pour ce faire, ils cherchent à reconstruire le champ de distance de chaque particule prise individuellement

$$\phi(\mathrm{x}) = |\mathrm{x} - \bar{\mathrm{x}}| - \bar{r},$$

où $\bar{x}$ et $\bar{r}$ sont des estimations moyennées de la position et du rayon de particules voisines.

Il est alors nécessaire que le rayon de chaque particule soit une évaluation correcte de la distance à la surface. Adams *et al.* [APKG07] reprennent l'idée en considérant toujours des voisinages de particules et en évaluant plus finement les distances moyennes à la surface par une approximation de l'axe médian et une mesure locale des caractéristiques de la surface recherchée.

La fonction implicite caractérise généralement une information de distance. L'informatique étant fondamentalement un monde discret, la fonction est échantillonnée en un nombre fini de points de l'espace qui sont les sommets d'une structure de découpage spatial. La problématique devient ensuite l'extraction de l'information souhaitée.

Une des premières utilisation des surfaces implicites, due à Blinn [Bli82], utilisait un rendu par *ray tracing* et ne nécessitait pas d'extraction explicite de surface. Les processus de rendu actuels reposent principalement sur des maillages, donc une représentation explicite. Dans ce contexte, une fonction implicite n'est pas directement visualisable. Il est donc nécessaire d'extraire une isosurface à partir de la donnée de la fonction implicite.

Les méthodes d'extraction reposent fondamentalement sur la supposition que la fonction implicite est continue. On peut alors utiliser le théorème des valeurs intermédiaires :

Théorème 3.1.1 (Théorème des valeurs intermédiaires) [3]

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction continue sur un intervalle I.

Alors pour tous réels a et b de I, pour tout réel $k \in [f(a); f(b)]$, $\exists c \in [a; b]$ tel que $f(c) = k$.[4]

qui se généralise aux dimensions supérieures et en particulier aux fonctions de $\mathbb{R}^3$ dans $\mathbb{R}$.

L'idée principale est donc de considérer que si les valeurs des extrémités d'un élément de la grille implicite sont de part et d'autre de l'isovaleur recherchée, alors l'élément est coupé par l'isosurface.

L'algorithme du *marching cubes* [LC87] est le premier algorithme à avoir utiliser une approche tridimensionnelle dans le domaine de l'extraction d'isosurface. La surface est reconstruite en parcourant les voxels constituant la grille implicite. Chaque sommet est inférieur ou supérieur à l'isovaleur recherchée. Pour chaque voxel, en prenant la valeur de chacun des 8 sommets le constituant, on peut alors déterminer une configuration parmi les $2^8 = 256$ possibles. En considérant des jeux de symétrie dans le voxel, on peut se ramener à 15 configurations distinctes (fig. 3.2). La géométrie à créer est généralement déterminée par interpolation linéaire. La topologie associée dépend uniquement de la configuration du voxel et est recréée à partir d'une table synthétisant l'ensemble des cas possibles.

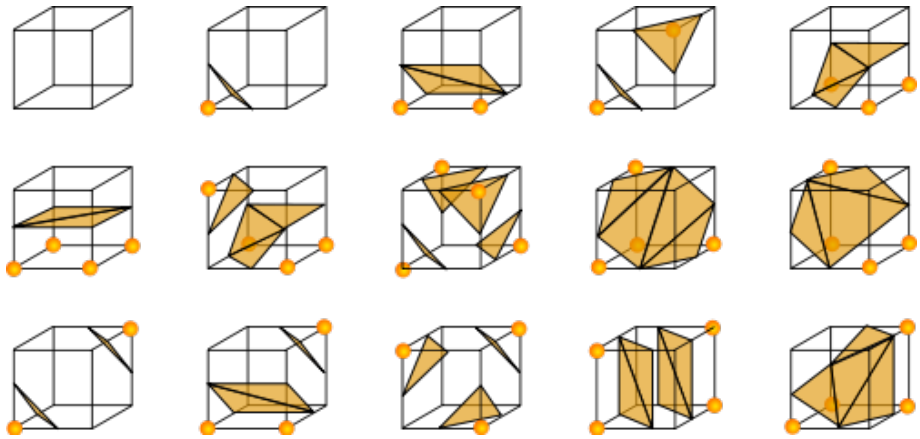


FIG. 3.2: Table de création des triangles pour *marching cubes*.

Lorsque les sommets diagonalement opposés d'une face de la grille implicite sont de même signe et de signe distinct par rapport aux sommets voisins, il y a ambiguïté sur la topologie à reconstruire [vGW94] (fig. 3.3). Du fait de l'indépendance entre les voxels, il peut arriver que la topologie recréée ne soit pas consistante entre des voxels voisins. Triquet *et al.* [TMC01] proposent une solution basée sur le fait que les arêtes générées dans les cas ambigus doivent appartenir à deux triangles exactement. Avec une intervention utilisateur, ils construisent une surface cohérente mais dont la topologie n'est pas forcément la topologie réelle de la surface. Lewiner *et*

[3] Pratiquement, on utilise très souvent le théorème sous la forme :

Théorème 3.1.2 (Théorème de Bolzano) *Si $f(a) \leq 0$ et $f(b) \geq 0$, alors $\exists c \in [a; b]$ tel que $f(c) = 0$.*

[4] Il est à noter que le théorème donne l'existence d'au moins un c dans l'intervalle $[a; b]$ mais ne spécifie rien sur son unicité. Concrètement, dans notre cas, il pourrait y avoir plusieurs passages de l'isosurface donnant des repliements de surface mais ils ne seront pas détectés.

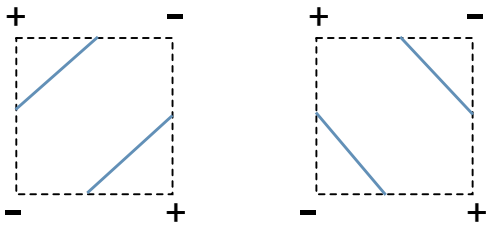


FIG. 3.3: Exemple 2D d'échantillonnage de fonction implicite ambigu.

al. [LLVT03] corrigent le problème de façon automatique en étendant la liste des configurations initiales de voxels du *marching cubes* avec des sous-configurations prenant en compte les configurations des voxels voisins. Ceci permet de résoudre complètement les ambiguïtés et de reconstruire la topologie exacte de la surface implicite.

Guéziec et Hummel considèrent un découpage de l'espace non pas en parallélépipèdes rectangles mais en tétraèdres et proposent ainsi l'approche *marching tetrahedra* [GH95]. Chaque parallélépipède de la grille du *marching cubes* peut simplement être découpée en six tétraèdres. L'utilisation de volumes élémentaires plus simples lève directement les problèmes de topologies. De plus, cela permet également de créer des triangles mieux proportionnés.

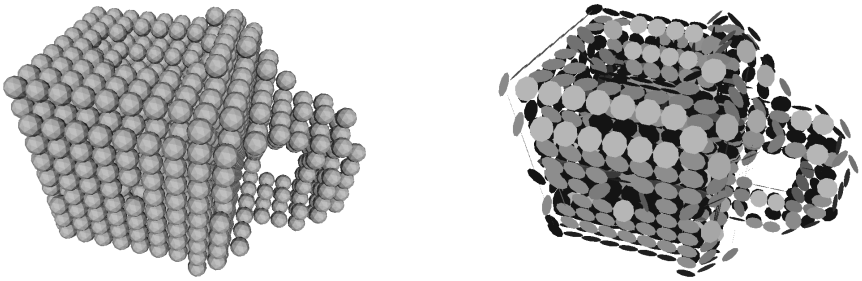
Au lieu de considérer des configurations de voxel spécifiques, Ho *et al.* [HWC+05] proposent de construire la topologie sur la base des faces des voxels de la grille implicite. La reconstruction passe donc par une étape en 2D de reconstruction des arêtes pour prendre en compte les cas pathologiques du *marching cubes*. Via une heuristique utilisant les informations de normale aux sommets, les caractéristiques de la surface sous-jacente sont conservées, sans introduire de dépendance forte entre des voxels voisins. Comme chaque surface de voxel est partagée exactement par deux voxels, l'utilisation d'une structure adaptative n'engendre pas de trou indésirable lors de la reconstruction.

Les approches primales se limitent à positionner la géométrie sur les arêtes de la grille implicite. Les positions des sommets de la surface sont donc fortement contraintes. Les approches duales proposent de donner plus de liberté en positionnant la géométrie à l'intérieur des voxels. Ju *et al.* [JLSW02] proposent une approche duale basée sur l'approche du *marching cubes* et *extended marching cubes*. Dans un premier temps, des données hermitiennes *i.e.* des positions munies de normales sont déterminées par l'algorithme du *marching cubes*. Pour chaque voxel ayant un changement de signe, un sommet minimisant une énergie quadratique basée sur les données hermitiennes est calculé. La topologie est créée en reliant les sommets des quatre voxels partageant une arête à changement de signe. En prenant en compte les arêtes « minimales » , *i.e.* les arêtes des voxels feuille, il est possible de reconstruire la surface à partir de structures adaptatives.

Schaefer et Warren [SW04] proposent une approche mixte alliant méthode primale et méthode duale. Le principe de l'approche consiste à construire une grille duale permettant de représenter une fonction implicite par un octree, de façon à avoir une représentation compacte en fonction d'une précision donnée. La géométrie de la grille duale est construite par minimisation d'une erreur quadratique dans les feuilles de l'octree puis la topologie est construite récursivement à partir de la géométrie et de la topologie de l'octree. Chaque cellule de la grille duale ainsi générée est vue comme topologiquement équivalent à un voxel parallélépipédique rectangle et peut donc être traitée selon le contourage du *marching cubes*.

Discussion

L'utilisation d'approches discrètes donne des résultats peu satisfaisants (fig. 3.4).



(a) Visualisation par primitives sphériques. (b) Visualisation par *surfels*.

FIG. 3.4: Visualisations discrètes d'un système de particules.

La visualisation par primitives simples ne permet pas de percevoir de façon satisfaisante l'ensemble du volume (fig. 3.4a). La visualisation par *surfels* pose plusieurs problèmes (fig. 3.4b). Dans un premier temps, il faut que l'évaluation de la normale à chaque particule soit précise sous peine d'avoir des disques mal orientés. L'utilisation du schéma des différences finies centrées donne clairement de meilleurs résultats qu'une utilisation de *SPH* pour l'évaluation du gradient mais cela implique un coût d'au moins 6 évaluations de fonction implicite par *surfel*. L'utilisation de particules orientées délocaliserait le surcoût dans les calculs de simulation.

Même si des *shaders* spécifiques peuvent améliorer les résultats, l'utilisation de *surfel* est peu appropriée à notre cas.

La reconstruction de surface sur nuage de points est un domaine bien étudié et qui peut être utilisé dans notre contexte puisque le système de particules peut être vu comme un nuage de points. L'objectif est d'extraire une surface englobant « au mieux » le nuage de points. Dans la mesure où il est difficile de trouver une bonne métrique pour juger de la qualité de la surface obtenue par rapport à un nuage de points, nous chercherons à obtenir un résultat subjectivement plaisant.

La reconstruction d'une surface lisse par morceaux [BRS04] est intéressante, notamment car elle permet une reconstruction en parallèle. Cependant, il faut un nombre relativement important de points sur ou à proximité de la surface pour espérer obtenir de bons résultats puisque les sommets du maillage de contrôle d'un *patch* sont uniquement les points donnés en entrée. De plus, la nature « par morceaux » de la surface reconstruite peut devenir un inconvénient pour effectuer des post-traitements sur la surface.

L'approche par fronts [SLS+06] pourrait tirer parti des informations volumiques pour la phase d'expansion des fronts qui est basée sur un champ de distance. Mais le surcoût induit notamment par le lissage du front et la gestion des collisions inter-fronts font que l'approche semble moins adaptée à notre besoin de reconstruction en temps réel qu'une approche purement implicite.

Les approches de reconstruction directe de surface sur nuage de points permettent d'obtenir une représentation fidèle de la surface sous-jacente au nuage en utilisant le fait que le nuage est généralement supposé proche de la surface. Dans notre cas, le nuage de points est volumique donc la proportion de points proches

de la surface est généralement relativement faible. Il faudra donc ajouter à ces méthodes un pré-traitement sur le nuage de points pour détecter ceux susceptibles d'être proches de la frontière. De plus, il faudrait ajouter une gestion des paramètres de particules puisque les approches existantes utilisent la notion de point au sens mathématique du terme, *i.e.* un objet sans dimension, alors que nos particules ont, en plus d'une position, un ensemble de paramètres physiques.

Les approches implicites sont donc intéressantes dans ce contexte puisqu'il est possible de définir n'importe quelle fonction implicite. La fonction définie doit seulement être continue. Il va donc s'agir de définir une fonction implicite permettant de caractériser le nuage de matière en prenant en compte les propriétés physiques des particules. Il sera nécessaire de s'interroger sur l'interprétation à donner au nuage de particules à l'instar des approches construisant un champ de distance.

L'évaluation discrète de la fonction implicite peut se faire sur une grille régulière ou sur une grille adaptative, l'intérêt étant alors de limiter les ressources utilisées. L'inconvénient des structures hiérarchiques réside dans l'accès aux éléments qui nécessite un parcours d'arbre. De plus, nous utilisons un matériau homogène. Il semble alors préférable de chercher à fournir une surface reproduisant au mieux cette caractéristique et donc d'utiliser une grille régulière pour l'échantillonnage de la fonction implicite.

Une fois le choix de fonction implicite déterminé, il est nécessaire de s'interroger sur la méthode utilisée pour extraire une surface. Les méthodes primales, *i.e.* générant la géométrie sur les arêtes de la grille implicite, sont simples à mettre en place et nécessitent peu de calculs. Leur défaut principal est de contraindre fortement la géométrie. La finesse du détail capté est, dans une mesure plus importante que pour les approches duales, fortement lié à la résolution de la grille implicite utilisée. Les approches duales sont plus souples en permettant un positionnement de la géométrie à l'intérieur des voxels de la grille implicite. Cependant, le calcul des positions des sommets nécessite alors une minimisation d'une énergie quadratique dont le coût est non négligeable pour une reconstruction en temps réel.

Dans la mesure où l'objectif recherché est celui d'un habillage plausible et plaisant sur le nuage de particules, le défaut des approches primales est minimisé. De plus les approches primales sont, par nature, très parallélisables puisque les voxels sont indépendants les uns des autres. Aussi, pour répondre à la contrainte forte de rapidité de la reconstruction, nous utiliserons une approche primale basée sur *marching cubes*.

Ce choix nécessite une réflexion sur la pertinence de la topologie reconstituée. Nous avons vu que le *marching cubes* pouvait générer des configurations ambiguës pour lesquelles il est nécessaire d'avoir un post-traitement utilisant les informations des voxels voisins pour corriger les problèmes. Les *marching tetrahedra* permettent de corriger le problème intrinsèquement. Néanmoins, le nombre d'éléments à examiner est nettement supérieur et, la qualité de régularité des triangles vient au prix d'un nombre bien supérieur de triangles extraits.

Les méthodes de correction du *marching cubes* engendrent un léger surcoût mais n'altèrent pas la nature parallèle des approches primales puisque les informations sur les voxels adjacents sont accédées en lecture seule et servent uniquement à générer la topologie du voxel courant. Dans la mesure où l'on travaille sur de la modélisation de forme libre, il n'est pas possible d'assurer la non-existence des cas pathologiques. Nous avons toutefois pris le parti de ne pas les considérer explicitement en estimant que :

1. les cas ambigus sont, de façon générale, statistiquement rares [vGW94]

2. dans notre contexte, les cas ambigus sont éphémères : l'utilisateur manipule la matière de façon continue ce qui peut automatiquement corriger les problèmes ;

3. la surface extraite sera post-traitée et subira notamment une décimation qui pourra également traiter le problème.

Nous allons à présent détailler les choix que nous avons retenus pour l'habillage en temps réel du matériau. A l'instar de Triquet *et al.* [TMC01], nous ne présenterons pas une nouvelle approche en tant que telle mais nous présenterons des innovations liées à notre contexte d'utilisation des surfaces implicites.

3.2 Une approche physique de la surface

3.2.1 Densité de matière et surface

Contrairement aux nuages de points issus de *scanner* 3D, les nuages de particules que nous considérons représentent un volume de matière et il est délicat d'en extraire une définition de distance à la surface sous-jacente. De plus, les particules possèdent des caractéristiques physiques autres que leur position.

Il se peut également qu'au cours de la simulation quelques particules se retrouvent isolées ou légèrement éloignées de la « masse » de matière. Il semble clair que dans ce cas, on ne souhaitera pas visualiser ces informations. Nous souhaitons donc une modélisation physique et filtrante de la surface.

La définition de fonction implicite doit ainsi :

1. prendre en compte les propriétés physiques des particules [Des97] ;

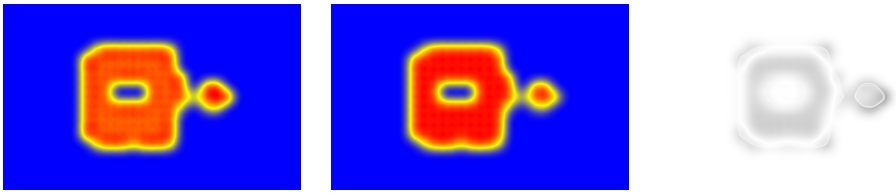
2. être locale *i.e.* la « manipulation » d'une particule ne doit modifier la surface que dans son voisinage ;

3. filtrer des particules isolées.

Les approches de reconstruction sur nuage volumique définissent souvent la fonction implicite comme la fonction caractéristique de la surface *i.e.* une fonction valant 1 pour les points situés à l'intérieur de la surface et 0 ailleurs, et font la supposition que toutes les particules sont dans l'intérieur du volume délimité par la surface à reconstruire (fig. 3.5a). Cette fonction, appelée *color field*, est facilement évaluable par la méthode *SPH*. L'inconvénient de cette définition vient du fait que chaque particule est considérée à l'intérieur de la surface, dès lors chaque particule est susceptible d'engendrer une création de surface au risque de créer une surface ondulée ou *blobby* et de créer des morceaux de surface parasites dans le cas de particules isolées. Elle ne satisfait donc pas notre deuxième critère.

Conceptuellement, la densité de matière est une grandeur physique, satisfaisant le premier point requis [DC99] (fig. 3.5b).

D'un point de vue computationnel, définir la fonction implicite uniquement comme étant la densité de matière est intéressant puisque le calcul de densité est le point de départ de tout calcul dans le formalisme *SPH*. De plus, les calculs effectués dans le formalisme *SPH* sont locaux puisque le noyau interpolant a un support compact. On répond ainsi au deuxième critère défini précédemment.



(a) *Color field.* (b) Densité (c) Différence

FIG. 3.5: Comparaison des fonctions implicites *color field* et du champ de densité.

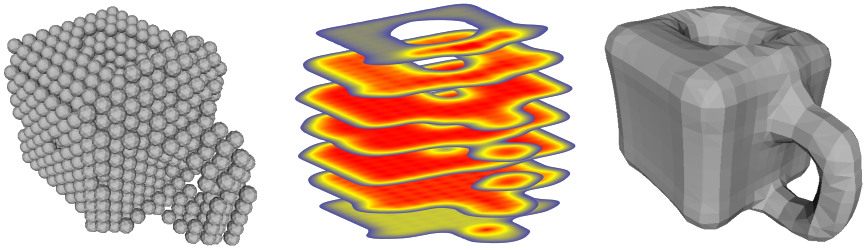


FIG. 3.6: Illustration de la reconstruction de surface sur le champ de densité.

La comparaison (fig. 3.5c) des fonctions implicites montre que le choix du champ de densité fournit un résultat visuellement plus lisse et plaisant.

Enfin, en jouant sur la valeur de densité extraite, un effet lissant est obtenu et il faudra plus ou moins de particules pour générer une surface. La variation d'isodensité extraite remplit donc le rôle de filtre souhaité. Il est cependant nécessaire de ne pas chercher à extraire une densité trop grande si l'on souhaite conserver une compréhension intuitive de la surface extraite puisque plus la densité désirée sera élevée, plus elle nécessitera une concentration importante de particules et donc la surface risque d'être générée « à l'intérieur » du nuage de particules plus qu'à sa périphérie.

Nous utilisons donc la densité de matière, calculée par la méthode *SPH*, comme fonction implicite. Ce choix permet de répondre aux trois critères nécessaires. Cette définition est, par ailleurs, cohérente avec notre approche physique des formes (fig. 3.5).

Un résultat fondamental lié aux fonctions implicites est que le gradient d'une fonction scalaire est orthogonal en tout point à la ligne de niveau de la fonction en ce point. Ce résultat permet donc de calculer formellement les normales aux sommets de la surface (ce qui est important pour les calculs d'éclairage de celle-ci). Originellement, Lorensen et Cline proposent d'utiliser des schémas de type différences finies pour effectuer ces calculs. La méthode *SPH* permettrait également de le faire. Nous avons cependant ici opter pour une détermination géométrique des normales : pour chaque triangle généré, une normale est calculée via un produit vectoriel. La normale de chaque sommet est la somme, pondérée par leur aire respective, des normales des triangles adjacents. L'étude réalisée par Jin *et al.* [JLW05] montre que les résultats ainsi obtenus, moins coûteux que les calculs exacts, sont accep-

tables. Dans la mesure où la surface construite a pour objectif d'être plaisante, et que visuellement les résultats le sont, nous avons validé cette approche.

3.2.2 Structures de données

Il est important de s'interroger sur la façon de stocker les données liées à la fonction implicite. Dans notre cas de matériau homogène, celle-ci est échantillonnée sur un ensemble discret de points réguliers. L'avantage d'utiliser une grille [DC99] est que l'accès aux données est direct et que la structure est relativement *cache friendly* (les processeurs modernes travaillent bien plus efficacement si les données en mémoire sont proches spatialement). L'inconvénient majeur est la nature statique de la structure qui nécessite de prévoir un espace suffisamment grand en introduisant au passage une contrainte forte pour l'utilisateur.

Il existe plusieurs solutions pour remédier au problème. La première est l'utilisation de structures arborescentes. En imposant une profondeur constante pour obtenir une grille régulière, il est par exemple possible d'utiliser un octree pour stocker les informations uniquement où c'est utile. L'inconvénient de l'octree est l'accès non direct aux données stockées. Ferley *et al.* [FCG00] proposent d'utiliser un arbre binaire équilibré qui a les mêmes propriétés qu'un octree.

Rosenberg et Birdwell [RB08] reprennent l'approche décrite par Cline *et al.* [CLL+88] et proposent une approche ne nécessitant le stockage — temporaire — que d'une tranche de la grille en combinant l'évaluation de la fonction implicite et l'extraction de l'isosurface désirée. Un découpage de l'espace en sous-volumes est également possible de façon à fournir une borne supérieure de mémoire nécessaire pour la reconstruction. Les tranches sont calculées les unes après les autres et un *buffer* de la tranche précédente est conservé pour avoir toutes les informations nécessaires au traitement d'une couche de voxels. L'inconvénient est que la fonction implicite est recalculée et la surface reconstruite entièrement à chaque *frame*.

Dans notre cas, en général, seule une partie de la surface sera à mettre à jour sur modification locale de l'utilisateur. De plus, comme nous le verrons en sous-section 3.2.4, des traitements sur la fonction implicite peuvent être envisagés. Notre contexte semble donc nécessiter un stockage complet et permanent de la fonction implicite.

Nous avons opté pour une table de hachage à l'instar de [FCG00, GCS99]. A un triplet d'entiers $(i, j, k) \in \mathbb{R}^3$ nous associons une grandeur flottante représentant la densité en ce point de l'espace. Dans le cas où, lors de la reconstruction, on souhaiterait accéder à un point où la densité n'a pas été calculée, la structure renverra une densité nulle signifiant l'absence de matière. De cette manière, nous avons une solution permettant :

- le stockage des informations uniquement où c'est nécessaire ;
- l'accès direct aux valeurs de la grille implicite.

3.2.3 Extraction locale

3.2.3.1 D'une construction de surface...

Originellement, Lorensen et Cline [LC87] ne séparent pas l'étape de construction de la géométrie *i.e.* la détermination des sommets du maillage, et la construction de la topologie *i.e.* la création des triangles reliant les sommets les uns aux autres.

Ils procèdent voxel par voxel et créent des points et des triangles pour chaque voxel. Ceci fournit, au final, une soupe de triangles et n'est pas souhaitable. En effet, chaque sommet de la grille implicite est partagé par huit voxels et chaque arête est partagée par quatre voxels de la grille implicite. Une approche naïve recalculant toutes les données pour chaque voxel sera donc inutilement coûteuse. De plus, une soupe de triangles est peu utilisable pour des algorithmes de post-traitement.

Le problème est levé en séparant bien les étapes [TMC01, FCG00, LLVT03] :

1. la fonction implicite est construite, une fois pour toutes, en évaluant la densité dans le volume défini par le système de particules ;

2. la géométrie est déterminée par le calcul des intersections de l'isosurface avec les arêtes de la grille ;

3. la topologie est reconstruite en utilisant des tables configuration indiquant les triangles à créer.

Pour associer les indices de sommets créés aux arêtes de la grille implicite, nous utilisons une table de hachage associant l'indice de l'arête, calculé grâce au triplet de l'extrémité inférieure de l'arête et d'une direction canonique, à l'indice de sommet. Pour accélérer certains traitements, on utilise également une table de hachage inverse, associant à chaque indice de sommet, un indice d'arête.

3.2.3.2 ... à une reconstruction semi-locale...

Le coût de calcul complet de la surface est important du fait du coût de calcul de la fonction implicite relativement élevé. Il nécessite une recherche de voisinage spatial pour chaque valeur implicite. Il convient donc de minimiser, autant que faire se peut, le nombre d'évaluations de la fonction implicite.

Müller *et al.* [MCG03] commencent par déterminer les particules situées à proximité de la surface en calculant la norme de la normale du champ de couleur en chaque particule. Cette approche est semblable à celle proposée par Hoppe [HDD+92] qui ne parcourt les voxels qu'à proximité du nuage de points en entrée pour minimiser le nombre d'évaluations de la fonction implicite.

Par ailleurs, les phénomènes physiques simulés sont continus, aussi la surface doit évoluer avec une certaine continuité. On peut donc faire la supposition qu'entre deux reconstructions successives, l'isosurface passera par des voxels voisins [Des97, DC99]. Concrètement, on peut supposer que si la surface traverse un voxel à l'instant t, elle traversera un des voxels du 1-voisinage à l'instant $t + dt$. Pour obtenir un résultat plus fin, il sera également possible de calculer le plus grand déplacement effectué par une particule et, en prenant la résolution de la grille implicite, d'en déduire le k-voisinage à considérer.

Cette approche par voisinage de voxels permet de s'affranchir de « parcours intelligents » des voxels [TMC01, RB08] et permet une implémentation efficace puisque les voxels sont complètement indépendants les uns des autres alors que le parcours par couche *marching slice* proposé par Rosenberg et Birdwell implique une corrélation entre les voxels.

Lors de la mise à jour de la surface, on commence donc par repérer la liste des voxels qui étaient traversés par l'isosurface lors de la reconstruction précédente. On détermine ensuite le k-voisinage de voxels de cet ensemble. La fonction implicite est

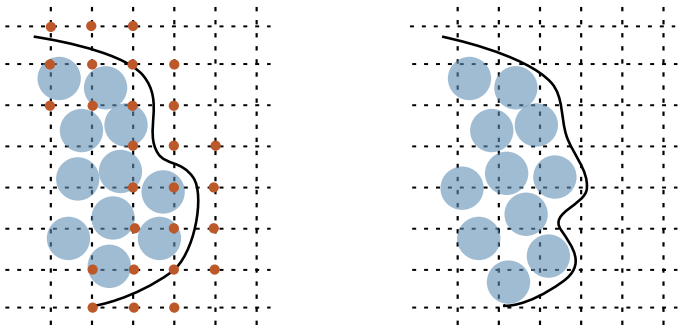


FIG. 3.7: Reconstruction semi-locale de surface implicite par échantillonnage à proximité de la surface.

uniquement mise à jour aux sommets de ces voxels et la surface est reconstruite en parcourant seulement ces voxels mis à jour (fig. 3.7).

Si cette approche permet une économie du nombre d'évaluations de la fonction implicite, elle n'est pas pleinement satisfaisante puisqu'elle ne permet pas de stocker la fonction implicite dans son ensemble, ce qui peut être nécessaire pour des traitements comme nous le verrons en 3.2.4 et surtout, elle nécessite un recalcul complet de la surface à extraire. L'approche est efficace dans le cas où l'ensemble du système est modifié *e.g.* lorsque des comportements de matériaux sont ajoutés à chaque particule.

Nous allons donc chercher une méthode de reconstruction strictement locale.

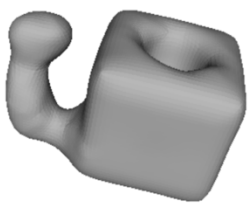
3.2.3.3 ... à une reconstruction locale

Localiser la mise à jour de la fonction implicite pour minimiser le coût des calculs est trivial. Seuls les échantillons de la fonction à proximité de matière modifiée sont calculés. Dans notre cas, les modifications de la matière dues à l'utilisateur peuvent être très locales. Il serait donc intéressant de localiser finement la reconstruction de la surface.

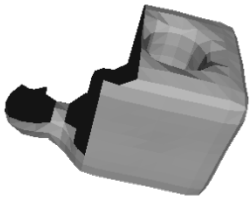
Initialement, Lorensen et Cline proposent une construction par soupe de triangles. La mise à jour locale est alors facile puisqu'il suffit de supprimer les triangles situés dans un voxel et de déterminer, après mise à jour du voxel, les triangles à créer à l'intérieur du voxel modifié, et ce indépendamment des autres données.

Plus généralement, l'approche du *marching cubes* est intrinsèquement locale puisque l'extraction de la topologie est effectuée en prenant en compte uniquement l'information d'un voxel. En utilisant le fait que la géométrie de la surface est rattachée aux arêtes de la grille implicite et que la topologie est, elle, rattachée aux voxels de la grille implicite, il devient possible d'extraire et de mettre à jour localement la surface implicite plutôt que de construire une soupe de triangles ou de reconstruire globalement la surface.

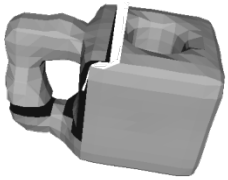
Un sommet d'une grille supposée infinie est partagé exactement par six arêtes en tant qu'origine ou extrémité d'une arête dans chacune des directions canoniques. Chaque arête de la grille est à son tour partagée par quatre voxels. Par conséquent, une modification de valeur de la grille implicite va engendrer six calculs d'intersection de l'isosurface avec la grille pour mettre à jour localement la géométrie et huit analyses de configuration de voxel (fig. 3.8).



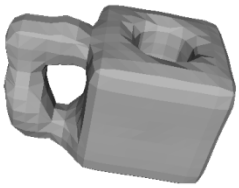
(a) Surface avant ajout de matière.



(b) Découpe du morceau de surface à reconstruire.



(c) Mise à jour du morceau à reconstruire.



(d) Recollement avec le reste de la surface.

FIG. 3.8: Reconstruction locale basée sur une modélisation par voxel.

Il est à noter qu'une modélisation du *marching cubes* uniquement basée sur les voxels permet une mise à jour locale mais celle-ci sera plus coûteuse qu'une modélisation par arête pour la géométrie et par voxel pour la topologie. En effet, avec une modélisation par voxels, la mise à jour d'un sommet de voxel implicite entraîne la nécessité de mettre à jour l'ensemble du voxel ce qui signifie supprimer toute la géométrie qu'il référence. Les triangles reliant le voxel aux voxels voisins non modifiés sont alors supprimés et il devient nécessaire d'effectuer une reconstruction en deux étapes. La première étape reconstruit la surface dans les voxels modifiés et la seconde permet de recoller le morceau de surface reconstruite au reste de la surface. Le traitement est inutilement alourdi (fig. 3.8).

La mise à jour d'un sommet de la grille implicite peut engendrer un changement de configuration de voxel par conséquent, il est nécessaire de supprimer toute la topologie d'un voxel dont un sommet est modifié. Par contre, seule la géométrie des arêtes impactées par une modification doit être supprimée (fig. 3.10).

L'algorithme de (re)construction de surface est donné par l'algorithme 2.

Cet algorithme de reconstruction de la surface est donc local (fig. 3.9) et permet la prise en compte de la localité des interactions utilisateur.

Le processus de reconstruction découle fortement de la fonction implicite. Notre choix du champ de densité implique une recherche de voisinage par valeur de la fonction implicite. Nous allons à présent fournir des pistes pour améliorer les temps de reconstruction en modifiant l'évaluation de la fonction implicite.

3.2.4 Échantillonnage

Le traitement informatique des surfaces implicites passe par un échantillonnage de la fonction implicite traitée. Dans notre contexte, nous avons choisi un échan-

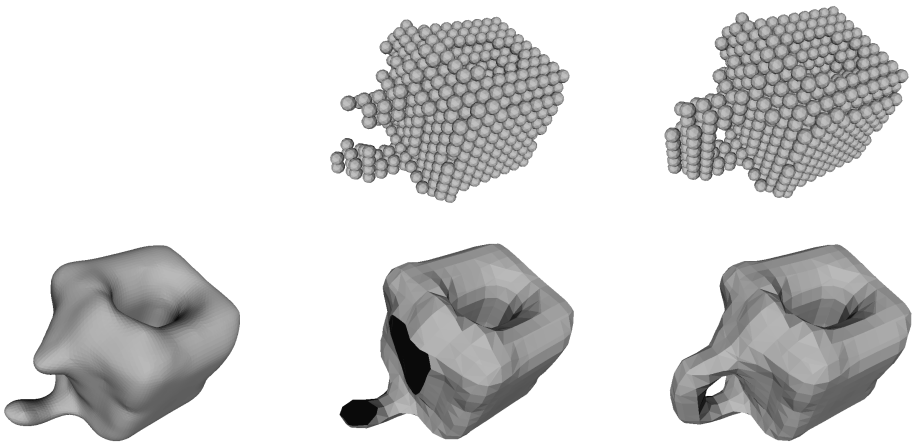


FIG. 3.9: Reconstruction locale avec mise à jour de la géométrie basée sur les arêtes.

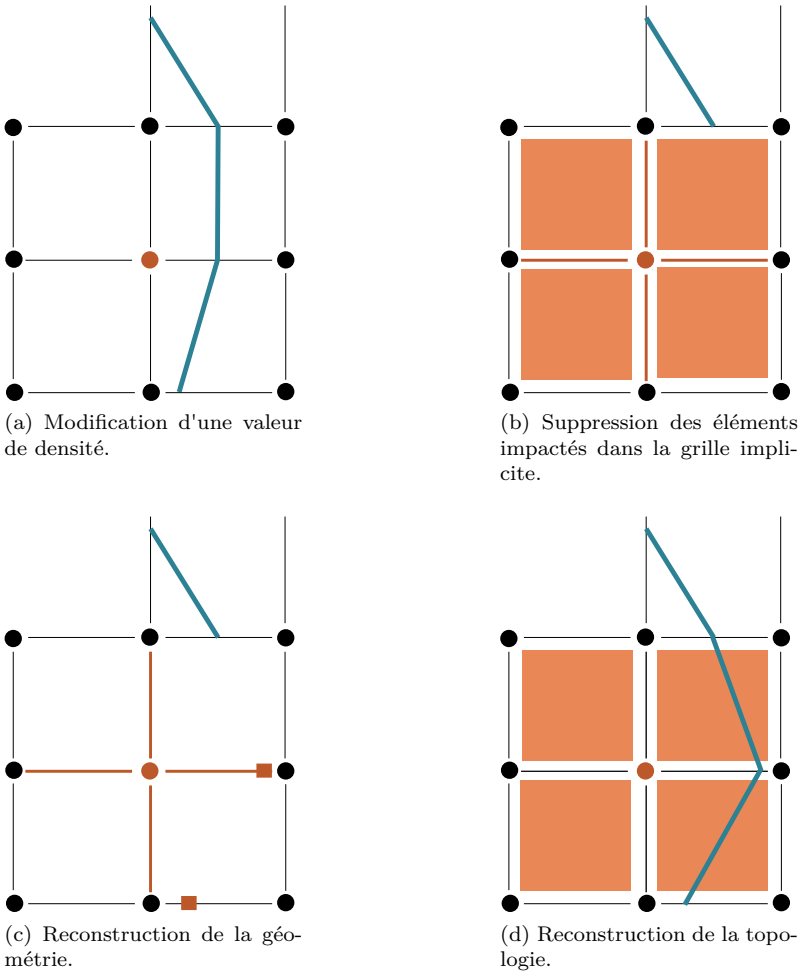
Algorithme 2 Mise à jour locale pour la méthode des *marching cubes*

déterminer les arêtes de la grille implicite mises à jour ;
déterminer les voxels de la grille implicite mis à jour (fig. 3.10a) ;
pour tous les voxels modifiés **faire**
  supprimer les triangles existants (fig. 3.10b)
fin pour
pour toutes les arêtes modifiées **faire**
  supprimer les sommets existants (fig. 3.10b)
fin pour
pour toutes les arêtes modifiées **faire**
  calculer les nouveaux sommets (fig. 3.10c)
fin pour
pour tous les voxels modifiées **faire**
  calculer les nouveaux triangles (fig. 3.10d)
fin pour

---

tillonnage régulier. La donnée de fonction implicite consiste ainsi schématiquement dans un tableau tridimensionnel de valeurs flottantes. Il semble ainsi assez naturel de vouloir faire un parallèle avec un ensemble de couches d'images ou un ensemble d'images possédant une profondeur [GCS99]. L'intérêt de ce parallèle est d'utiliser des outils du monde de l'image pour « améliorer » la qualité de la fonction implicite.

Dans la plupart des utilisations de surfaces implicites, la donnée de la fonction implicite est considérée comme une variable exacte ne devant être modifiée, et fixe dans le temps. Dans notre contexte, la fonction implicite est plus une interprétation temporaire du nuage de particules sous-jacent. Notre objectif final étant l'obtention d'une surface visuellement plaisante, nous pouvons nous permettre de modifier la fonction implicite calculée pour atteindre notre but. L'utilisation d'approches utilisées pour l'amélioration d'images est une piste. Par exemple, dans notre contexte, il serait envisageable d'utiliser des algorithmes de *blur* pour atténuer et lisser les contours sans perte réelle de détail [CLL+88].



(a) Modification d'une valeur de densité.

(b) Suppression des éléments impactés dans la grille implicite.

(c) Reconstruction de la géométrie.

(d) Reconstruction de la topologie.

FIG. 3.10: Influence de la modification d'une valeur implicite pour la reconstruction locale par *marching cubes*.

Un autre point majeur est celui de la performance. Celle-ci est grandement conditionnée par le nombre d'évaluations de la fonction implicite qui nécessite, dans notre cas, une recherche de voisinage spatial. Pour améliorer la performance en réduisant le nombre d'appels de calcul de densité, deux stratégies sont envisageables :

1. sur-échantillonner la fonction implicite en utilisant un schéma d'interpolation *i.e.* augmenter la résolution de la fonction implicite ;

2. sous-échantillonner la fonction implicite *i.e.* diminuer la résolution de la fonction implicite.

3.2.4.1 Sur-échantillonnage

Grisoni et Schlick [GCS99] mentionnent l'utilisation d'un ré-échantillonnage de la fonction implicite par interpolation linéaire mais rejettent l'utilisation du fait de la continuité généralement assurée de la fonction implicite. Leur contexte est cependant

différent puisque les auteurs manipulent un champ implicite déjà déterminé. Dans notre système, ce champ implicite est modifié en permanence et le coût de mise à jour de la fonction est le principal facteur limitant.

Galyean et Hughes [GH91] utilisent des sommes pondérées des valeurs de la fonction implicite pour lisser la surface générée par leur système.

En étendant cette idée, une solution peu coûteuse pour augmenter la résolution de la fonction implicite est d'interpoler les valeurs de certains sommets à partir de données exactes. Un schéma basé sur une interpolation linéaire peut être utilisé.

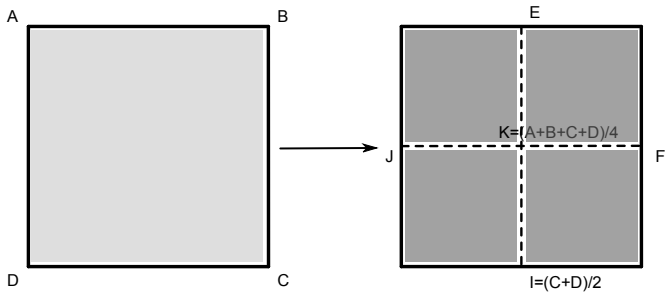


FIG. 3.11: Sur-échantillonnage de fonction implicite 2D par interpolation bilinéaire.

On peut ainsi doubler la résolution de la fonction implicite en utilisant uniquement les données des huit sommets d'un voxel et quelques moyennes (fig. 3.11). Pour ne pas gaspiller d'espace mémoire, le sur-échantillonnage peut n'être effectué que sur les voxels ayant un changement de signe.

Ce raffinement peut être appelé successivement à l'instar d'un schéma de subdivision de maillage (fig. 3.12).

3.2.4.2 Sous-échantillonnage

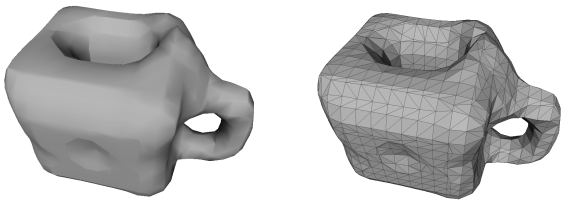
Diminuer le coût de l'évaluation de la fonction implicite peut également se faire simplement en évaluant la fonction implicite sur moins de points.

Labsik *et al.* [LHMG02] utilisent cette idée de façon à reconstruire un maillage initial par *marching cubes* avec le nombre le plus faible possible de polygones. En partant de la résolution la plus haute, ils diminuent de moitié la résolution en prenant pour chaque point de la fonction implicite dégradée, le maximum des valeurs des voxels compactés. Grisoni et Schlick [GCS99] utilisent un filtre basé sur la décomposition de Haar.

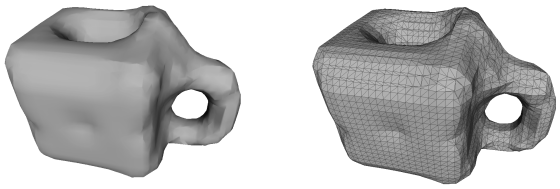
Ces filtres modifient les valeurs. La fonction implicite ne correspond donc plus à la fonction implicite qui serait calculée sur la base de la résolution plus faible. Une idée est donc de ne calculer et d'utiliser les valeurs de la fonction implicite sur un sous-ensemble de la résolution souhaitée en temps réel (fig. 3.13). Le reste de la fonction implicite et la surface correspondante peuvent être calculés lorsque les besoins de calculs autres sont réduits (fig. 3.14).

3.2.4.3 Discussion

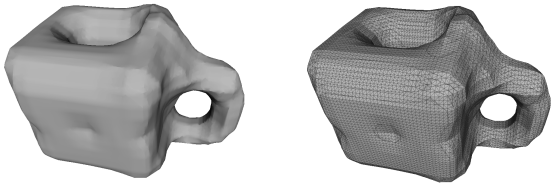
La qualité de la surface reconstruite dépend fortement de la résolution utilisée pour la fonction implicite. Plus celle-ci sera fine et meilleurs seront les résultats finaux. L'augmentation de la résolution au moyen d'une interpolation donne des résultats médiocres. Comme souligné par Grisoni et Schlick [GCS99], ceci peut s'expliquer



(a) Surface implicite sur-échantillonnée une fois.



(b) Surface implicite sur-échantillonnée deux fois.



(c) Surface implicite sur-échantillonnée trois fois.

FIG. 3.12: Résultats obtenus par sur-échantillonnage de fonction implicite.

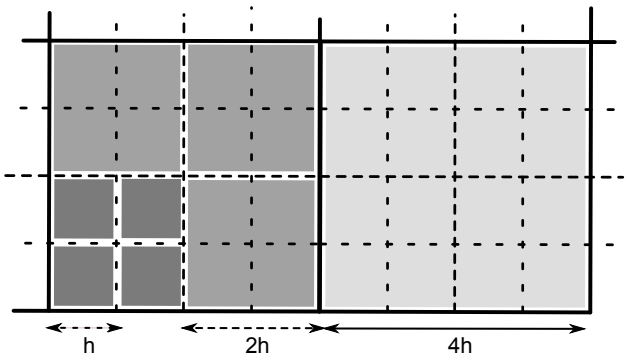
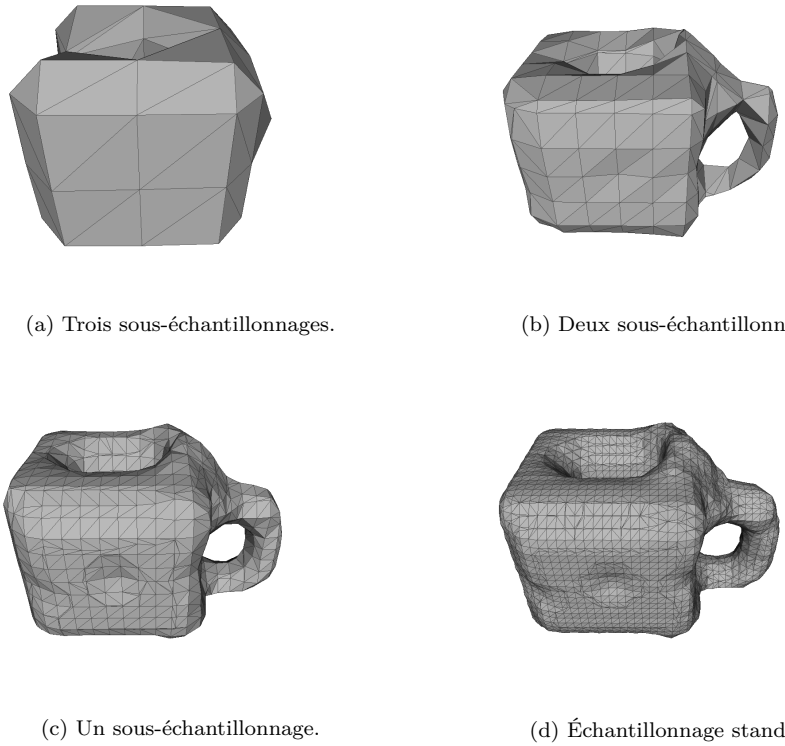


FIG. 3.13: Grilles 2D imbriquées utilisées pour le sous-échantillonnage. A chaque niveau, le nombre de valeurs implicites calculées et utilisées est divisé par deux dans chaque dimension.



(a) Trois sous-échantillonnages.

(b) Deux sous-échantillonnages.

(c) Un sous-échantillonnage.

(d) Échantillonnage standard.

FIG. 3.14: Résultats obtenus par sous-échantillonnage de fonction implicite.

par le fait que la fonction implicite calculée est $\mathcal{C}^1$ alors que l'interpolation linéaire utilisée ne fournit qu'une fonction $\mathcal{C}^0$. L'utilisation de schéma d'interpolation de meilleure qualité pourrait être envisagée mais le coût de calcul augmenterait. De plus, utiliser un niveau élevé de sur-échantillonnage déplace le coût de calcul de l'évaluation de la fonction implicite à son extraction. Chaque niveau multiplie approximativement par 4 le nombre de voxels coupés par l'isosurface ce qui devient rapidement non négligeable.

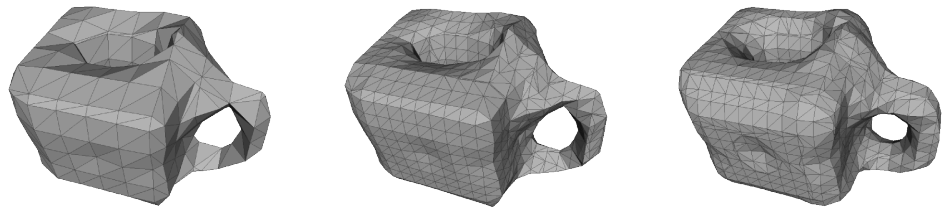


FIG. 3.15: Comparaison de stratégies d'échantillonnage de fonction implicite.

L'utilisation d'un sous-échantillonnage pour générer une surface de résolution moindre en temps réel est, elle, une solution intéressante. La visualisation ainsi fournie est légèrement dégradée pendant l'interaction utilisateur. On pourrait également envisager une reconstruction entièrement incrémentale à l'instar de la visualisation de données de cartographie interactives qu'on peut trouver aujourd'hui (fig. 3.14).

Il est également possible de coupler l'approche par sous-échantillonnage et par sur-échantillonnage ; on peut en effet utiliser une résolution faible artificiellement améliorée (voir fig. 3.15).

Notons que le choix du niveau de résolution retenu pourrait également être déterminé en fonction de la distance au point de vue. On pourrait ainsi étendre le principe du *mip mapping* [Wil83] à l'évaluation des fonctions implicites.

La modification de l'échantillonnage de la fonction implicite pour la reconstruction en temps réel fournit des résultats intéressants. Cependant, la qualité de la surface reconstruite reste visuellement assez polygonale et donc peu plaisante. Dans la mesure où le seul paramètre en amont de la reconstruction est la fonction implicite, dont nous souhaitons minimiser le nombre d'évaluations, nous allons voir comment améliorer la qualité de la surface reconstruite au moyen de post-traitements.

3.3 Vers une peau lisse

Si la peau reconstruite par surface implicite sur le nuage de particules semble habiller de façon plausible le volume défini par les particules, son apparence reste peu plaisante car trop grossière voire, contrairement à ce que l'on souhaite, peu organique. Il est donc nécessaire de se demander comment faire pour améliorer ces résultats.

L'utilisation de surfaces implicites, de par sa possibilité à manipuler n'importe quelle topologie de surface, et ce de façon locale, n'est pas en cause. Nous allons donc chercher à améliorer la surface obtenue en post-traitant les données.

3.3.1 Sur les bienfaits de la décimation

Notre objectif est l'amélioration subjective de la peau générée sur nos nuages de particules. La fonction implicite que nous utilisons est lisse. Aussi, l'aspect peu plaisant de la peau générée vient principalement de la difficulté d'avoir un échantillonnage important de la fonction implicite en temps réel et non pas de la présence de bruit.

Comme noté par Igarashi et Hughes [IH03], le *fairing* de surface a globalement un impact sur les hautes fréquences, fréquences qui ne sont globalement pas capturées par notre approche.

Par ailleurs, si un sommet de la grille implicite est proche de l'isovaleur extraite, il est probable que des triangles mal proportionnés, *i.e.* dont les longueurs d'arête sont très différentes, soient extraits. Labsik *et al.* [LHMG02] proposent d'utiliser une décimation en deux temps pour enlever ces triangles contenant peu d'information géométrique :

1. dans une première passe, les triangles dont les trois arêtes sont inférieures à un seuil (basé sur la résolution de la grille implicite) sont comprimés en un sommet,

2. puis, dans une seconde passe, les arêtes dont la longueur est inférieure au seuil sont également comprimées.

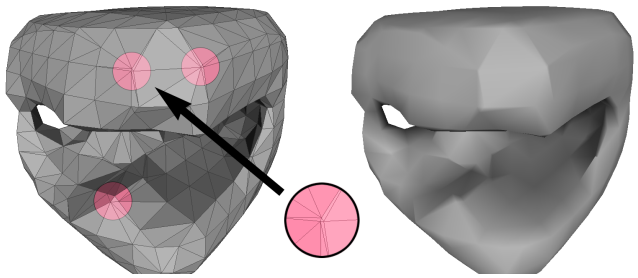


FIG. 3.16: Triangles pathologiques issus de l'extraction par *marching cubes*.

Ils constatent une réduction du nombre de triangles d'environ 20%.

À cette observation, on peut ajouter que la présence de cette géométrie peu significative est souvent associée à la présence de sommets extraordinaires (fig. 3.16).

Partant de ces constats, et dans la mesure où le seuillage utilisé par Labsik *et al.* [LHMG02] est subjectif, nous avons alors opté pour une approche plus directe. Nous avons ainsi utilisé une décimation basée sur la longueur d'arête uniquement et qui décime 30% des triangles de la surface extraite par *marching cubes*. Le sommet obtenu par compression d'une arête est positionné au barycentre des sommets de l'arête. L'avantage de ce type de décimation, comparé aux décimations basées sur des évaluations quadratiques locales de surfaces, comme introduites par Garland et Heckbert [GH97], est que les calculs mis en jeu sont nettement plus simples.

Les bénéfices de ce schéma de décimation sont multiples. Respectant le critère de décimation et l'observation de Labsik *et al.* , les triangles dégénérés sont supprimés. Comme nous avions observé que la présence des sommets extraordinaires était souvent liée à la présence de triangles dégénérés, la décimation a l'effet de bord souhaitable de diminuer fortement le nombre absolu de sommets extraordinaires ; la proportion des sommets extraordinaires change peu (fig. 3.17). On constate également que la décimation régularise la longueur d'arête et donne ainsi de « meilleurs » triangles dans le sens où les triangles seront proches de triangles équilatéraux. Enfin, en réduisant les petites arêtes qui sont à l'origine des fréquences les plus hautes, on produit une certaine tension de surface.

Si notre reconstruction de surface implicite est entièrement incrémentale, notre décimation prend en compte toute la surface. Comme notre objectif principal est la modélisation d'objets relativement petits et que la complexité de notre approche est linéaire, le coût de la décimation sur l'ensemble de la surface extraite est acceptable. Une décimation qui serait également incrémentale serait souhaitable.

La qualité visuelle est grandement améliorée sans réelle perte d'information géométrique. Notamment, la suppression des triangles dégénérés induit un ombrage de la surface plus plaisant (fig. 3.18). Cependant, celle-ci reste relativement anguleuse et nécessite un autre traitement. Nous avons opté pour une subdivision de Loop de façon à obtenir une surface lisse.

3.3.2 Subdivision de Loop efficace

La subdivision de Loop est basée sur les *box splines* à trois directions et est le schéma de subdivision le plus simple pour les maillages triangulaires. On renvoie à [ZS00, Bar05] pour une présentation générale du principe de subdivision, et

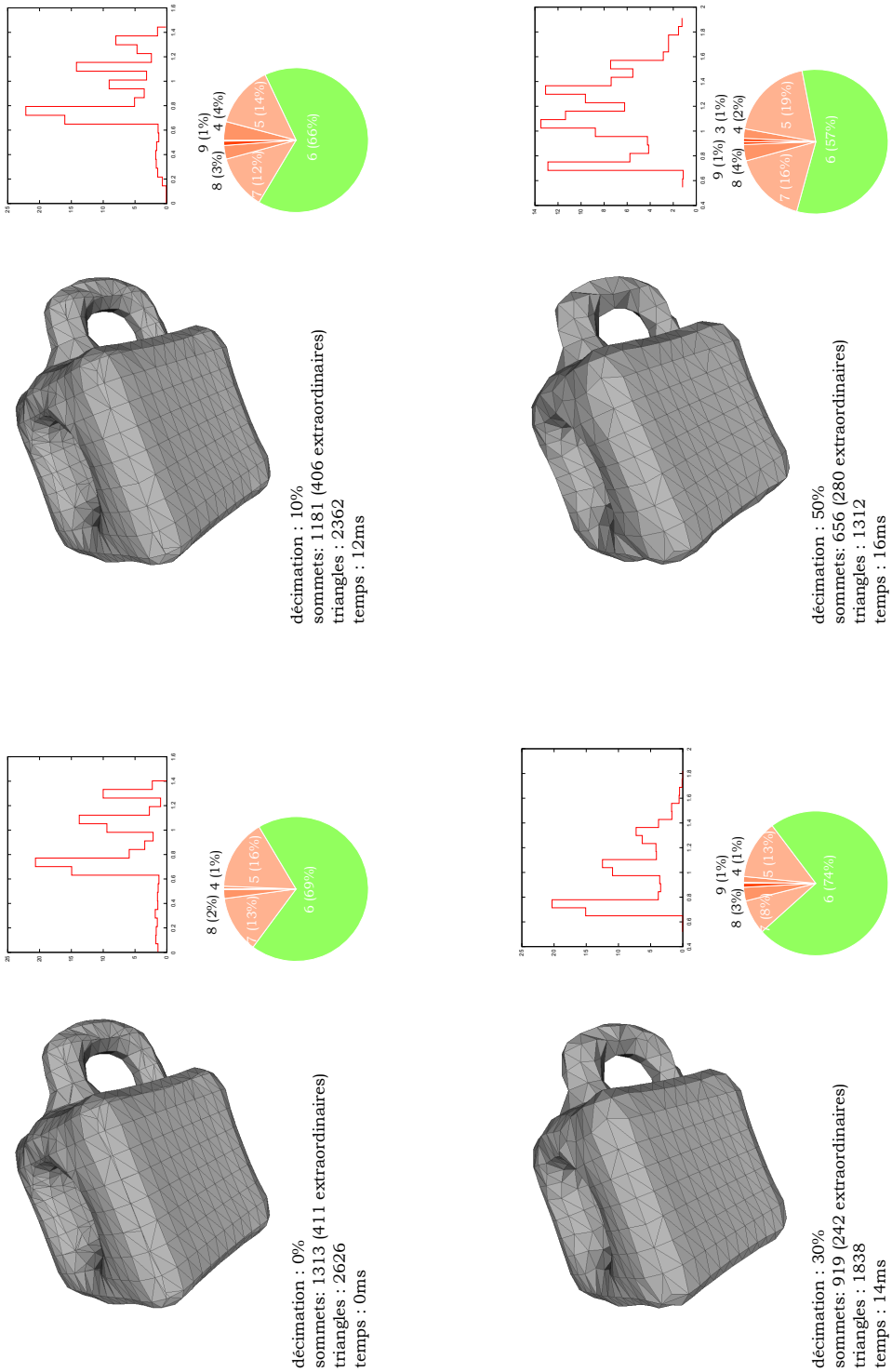


FIG. 3.17: Statistiques liées à la décimation (maillage, histogramme de répartition des longueurs d'arête, camembert illustrant les valences des sommets).

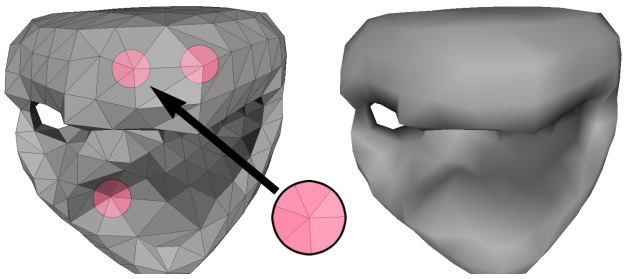


FIG. 3.18: Effet de la décimation sur les cas pathologiques.

à [Loo87] pour une présentation du schéma de Loop. Nous rappelons simplement que le schéma de Loop fournit une surface à continuité $\mathcal{C}^2$ aux sommets de valence 6 et de continuité $\mathcal{C}^1$ pour les sommets de valence extraordinaire.

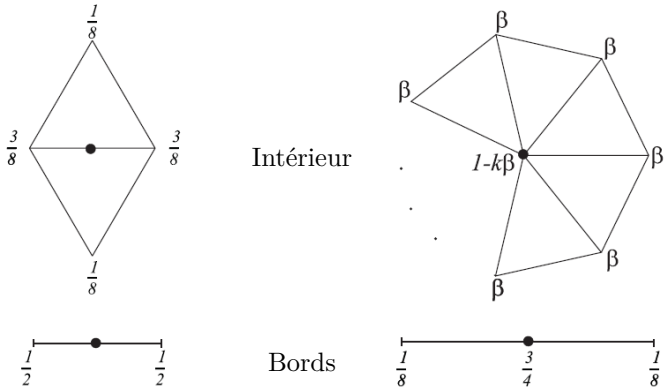


FIG. 3.19: Masque de subdivision de Loop

Les masques utilisés dans notre subdivision sont

$$\beta = \frac{3}{8k} \text{ pour } k > 3, \text{ et } \beta = \frac{3}{16} \text{ pour } k = 3,$$

comme préconisé par Warren [WW01] (fig. 3.19).

Si une implémentation naïve d'un schéma de subdivision est relativement simple, l'implémentation efficace est plus délicate. En effet, la subdivision nécessite un nombre important d'accès aux informations topologiques du maillage qui peut s'avérer coûteux.

Que ces informations topologiques soient stockées dans la structure de maillage [BPK+07] ou déterminées à la volée, les coûts d'accès aux données éparpillées en mémoire seront peu optimisés pour les architectures de processeur actuelles.

Au vu des premiers résultats obtenus avec une implémentation naïve, nous avons cherché une implémentation plus efficace.

L'accès à la topologie est très importante. Par nature, les schémas de subdivision ne font intervenir que des données relativement locales. Pour le schéma de Loop, chaque sommet n'a besoin que de l'information de topologie de son 1-voisinage. Les structures de données de maillage ne prennent pas en compte la proximité

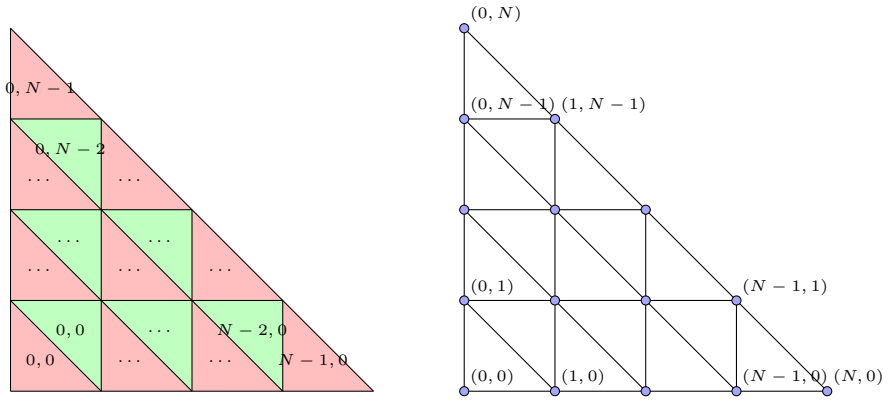


FIG. 3.20: Numérotation symbolique pour la subdivision de Loop.

topologique lors du stockage. Une recherche de voisinage peut donc nécessiter un parcours important des données en mémoire.

La littérature a déjà cherché à minimiser ces parcours en utilisant la notion de *patch* [PS96, Bri02]. L'approche présentée par Pulli et Segal [PS96] avait également pour but de limiter la consommation mémoire de la subdivision puisque à chaque niveau de subdivision, le nombre de triangles quadruple. Les auteurs proposent donc un appariement des triangles deux à deux et la création d'un *patch*, pour chaque paire, comprenant ces triangles ainsi que leur 1-voisinage. Un parcours du maillage en profondeur est ensuite effectué et le résultat de la subdivision d'un *patch* est rendu directement après son calcul. Les données sont alignées sur une grille de façon à pouvoir accéder aux informations de voisinage de façon implicite.

La problématique de consommation mémoire n'est plus réellement d'actualité sur les machines récentes et le coût de création des paires de triangles ne semble pas utile. Il semble plus approprié de constituer des *patchs* sur la base d'un unique triangle. L'avantage est qu'il n'y pas besoin de pré-processer le maillage d'entrée pour créer des appariements de triangles et l'approche serait également plus parallélisable (création d'un plus grand nombre de *patchs* contenant moins d'informations chacun). Nous avons donc écrit un algorithme reprenant les idées présentées par Brickhill [Bri02] et qui définit un *patch* comme un triangle du maillage de base augmenté de son 1-voisinage. On adapte l'agencement symbolique intelligent de Pulli et Segal [PS96] à l'utilisation d'un seul triangle et non plus de deux disposés selon un carré ; cette opération peut être vue comme un paramétrage symbolique de chaque triangle sur le triangle $\{(0,0),(2^n,0),(0,2^n)\}$ (où n est le niveau de subdivision).

Afin d'être le plus efficace possible au niveau du parcours de la mémoire, on utilise ensuite la bijection de $\mathbb{N}^2 \to \mathbb{N}$ suivante

$$u(i,j) = \frac{(i+j)(i+j+1)}{2} + j$$

qui nous permettra de stocker les données dans des tableaux unidimensionnels.

Parcourir le tableau des sommets revient à parcourir les sommets $(0,0)$, $(1,0)$, $(0,1)$, $(2,0)$, $(1,1)$, $(0,2)$, etc., c'est-à-dire en diagonale de droite à gauche, du bas vers le haut. Ainsi, le sommet $(0,0)$ se trouve en première position dans le tableau tandis que le sommet $(0,2^n)$ se trouve en dernière position.

Comme pour effectuer la subdivision d'un *patch* nous avons également besoin d'informations qui ne sont pas directement contenues dans le triangle, nous intro-

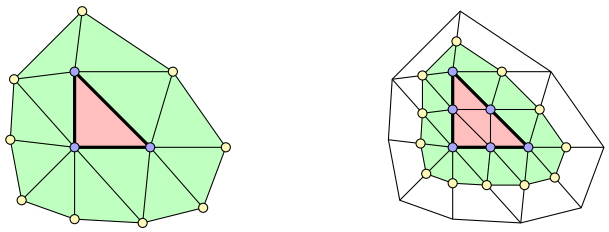


FIG. 3.21: *Patch* avant et après un niveau de subdivision dans le cas général.

duisons un peu de vocabulaire.

Soit v un sommet de maillage. On désignera par $|v|$ la valence de ce sommet.

À chaque triangle T d'un maillage, on associera une orientation ; ils seront toujours représentés dans le sens direct dans nos schémas. T peut donc s'écrire comme un triplet d'indices de sommets (v_i, v_j, v_k), à une permutation circulaire prés.

Dans un *patch* (fig. 3.21), on distinguera les sommets et triangles **réguliers** (*triangles rouges et sommets bleus*) des sommets et triangles **adjacents** (*triangles verts et sommets jaunes*). On dit aussi que ce sont les sommets ou triangles adjacents d'un *patch*.

On peut définir ces termes par induction. Le processus de subdivision d'un *patch* est donc :

- dans un *patch* de base, le triangle de base et ses sommets forment respectivement un triangle régulier et des sommets réguliers. Les autres triangles et sommets forment respectivement les triangles adjacents et les sommets adjacents (ou de support) ;
- si P est un *patch*, alors la subdivision de P se ramène à la subdivision des triangles réguliers, et des triangles adjacents de P ;
  - la subdivision des triangles réguliers est complètement conforme à la subdivision originale de Loop, où chaque triangle se divise en 4 triangles. Elle ne génère que des triangles et sommets réguliers, et déplace les anciens sommets réguliers.
    ▷ Un triangle régulier est un triangle issu de la subdivision d'un triangle régulier.
    ▷ Les sommets réguliers sont tous les sommets qui composent un triangle régulier, autrement dit tous les sommets à l'intérieur du *patch* ;
  - la subdivision d'un triangle adjacent ne génère que des triangles adjacents. Ce sont tous les triangles *dont l'un au moins* des sommets est un sommet régulier. Tous les nouveaux sommets générés qui ne sont pas réguliers sont les sommets adjacents.

Les sommets adjacents ne sont pas concernés par la numérotation précédente. Ils sont agencés différemment dans un bloc mémoire à la suite de celui réservé aux sommets réguliers. La disposition est plus simple, car l'irrégularité des valences de chaque sommet du maillage de base ne permet pas de trouver une bijection simple qui conserverait les informations de connexité des sommets.

Nous avons choisi de les stocker de telle sorte que parcourir ce tableau revient à parcourir les sommets dans le sens anti-horaire par rapport au triangle de base.

Les objets manipulés sont théoriquement des solides donc des surfaces sans bord. Cependant, les cas ambigus de la méthode *marching cubes* pourraient engendrer une surface à bords. Nous avons considéré toutes les configurations possibles pour un *patch*. Ces configurations E_iV_j, où i est le nombre d'arêtes bord et j le nombre de sommets bord ainsi que le dénombrement du nombre de sommets produits au cours de la subdivision, nécessaire pour l'allocation mémoire d'un *patch*, sont détaillés en annexe A.

Une fois l'agencement mémoire préparé pour le *patch*, le schéma de subdivision reste le même que dans l'implémentation standard. La seule différence est que les voisins de chaque sommet sont connus de façon implicite. Une fois tous les sommets calculés, les triangles sont déduits de façon triviale.

Il est à noter qu'il y a redondance des calculs au niveau des sommets introduits sur les arêtes partagées par deux triangles du maillage initial. Cette redondance de calcul, nécessaire au traitement des *patchs* indépendamment les uns des autres, pourrait introduire de légers artefacts visuels dus à la non commutativité des opérations flottantes. Cependant, dans la pratique, nous n'avons constaté aucune gêne visible.

La figure 3.22 présente quelques résultats obtenus sur un modèle de tasse sculptée avec notre système de pâte à modeler pour différents niveaux de subdivision. L'algorithme naïf et l'amélioration proposée ont la même complexité asymptotique. Pour chaque triangle initial, quatre nouveaux triangles sont créés et donc la complexité asymptotique est linéaire en nombre de triangles. Néanmoins, à mesure que le maillage se densifie, les accès mémoire aux informations topologiques deviennent de plus en coûteux. La courbe obtenue semble ainsi montrer une complexité asymptotique en $\mathcal{O}(n \log n)$. La gestion implicite des informations de topologie de notre approche lève les problèmes d'accès mémoire et la complexité théorique de l'algorithme est vérifiée par l'expérience. L'approche proposée ici est donc très efficace.

Ces résultats sont d'autant plus encourageants qu'ils ont été obtenus sans utiliser de parallélisme. Dans les deux cas, un seul processeur était utilisé. Il est donc clair que l'utilisation d'une numérotation symbolique est souhaitable pour obtenir des algorithmes très performants.

Dès un niveau de subdivision, l'implémentation proposée donne de meilleurs résultats que l'implémentation naïve, vraisemblablement grâce à une meilleure utilisation des mémoires *cache*.

Si notre implémentation est très efficace, il n'est cependant pas utile d'utiliser un niveau important de subdivision puisque les maillages de base sont relativement denses et afin de ne pas augmenter inutilement le temps de calcul du rendu. Aussi, nous utilisons un seul niveau de subdivision lorsque l'utilisateur édite le modèle et deux niveaux de subdivisions pour la visualisation sans modification de la surface.

3.4 Conclusion

Nous avons mis en place un schéma complet de traitement de la reconstruction d'une surface sur les nuages de particules. Les choix de la fonction implicite, de son échantillonnage, de l'extraction locale de la surface implicite et des différents traitements ont été réalisés dans un souci d'efficacité maximale pour une visualisation en temps réel.

Notamment, les solutions d'échantillonnage proposées sont évolutives et permettent de tirer avantage des architectures parallèles actuelles.

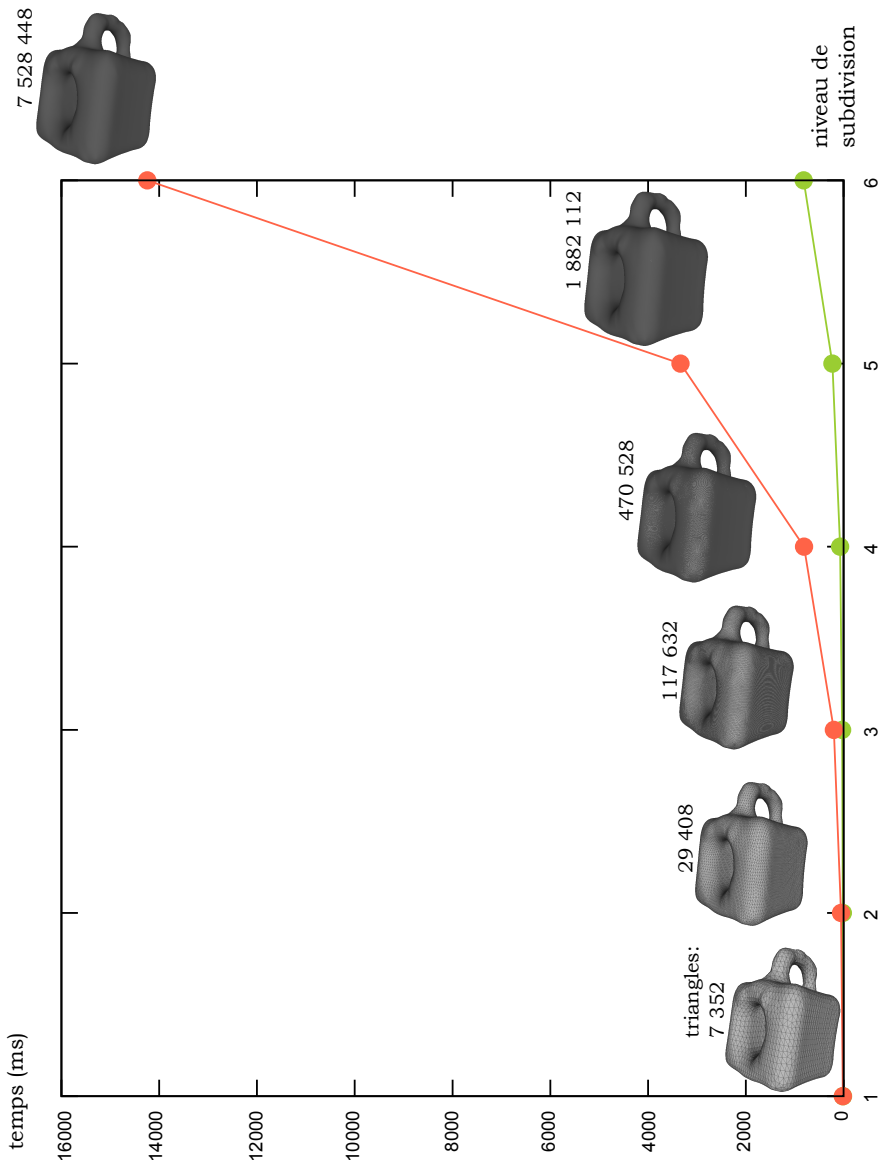
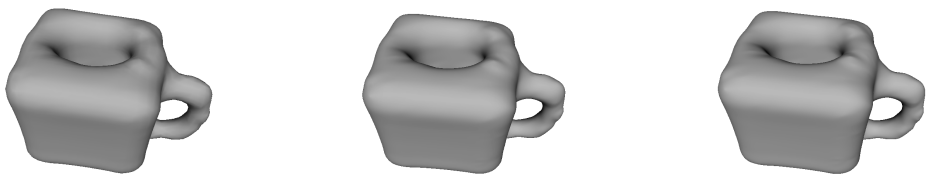


FIG. 3.22: Comparaison des temps de calcul pour l'implémentation naïve et l'implémentation *patchée*



(a) Un niveau de subdivision de Loop.

(b) Deux niveaux de subdivision de Loop.

(c) Trois niveaux de subdivision de Loop.

FIG. 3.23: Résultats obtenus par subdivision de Loop.

Les surfaces construites ont un aspect organique et lisse et correspondent au rendu souhaité pour des matériaux de type pâte à modeler.

De nouvelles techniques, basées sur des *shaders* modifiant le rendu pixel à l'écran, apparaissent et permettent de donner un aspect continu à un nuage volumique de particules sans extraire de surface [vdLGS09, Gre10, Sei09b]. Une difficulté soulevée par ces approches visuelles est la possibilité de générer une surface sauvegardable et réutilisable dans un autre contexte ; il faut pouvoir exporter dans une représentation standard comme un maillage, tout en minimisant les écarts entre la surface « visuelle » et la surface extraite mais en minimisant le nombre de primitives utilisées. La solution proposée dans ce chapitre est donc plus adaptée à notre contexte de manipulation de formes et non simplement de simulation.

Nous avons maintenant défini les fondements de l'utilisation de systèmes de particules pour la modélisation de formes. Il nous faut à présent réfléchir aux outils à proposer pour une manipulation intuitive.

Deuxième partie

Outils pour la sculpture

Outils pour la sculpture : Introduction

Nous avons défini un modèle de matériau pouvant avoir des comportements physiques, destiné à la création de formes ainsi que le processus de visualisation de ce matériau. Il convient à présent de réfléchir aux outils à proposer à l'utilisateur pour qu'il puisse créer du contenu 3D en se focalisant sur l'aspect artistique et créatif et sans se soucier des aspects techniques. Notre réflexion nous amène à considérer des outils les plus génériques possibles *i.e.* pouvant travailler sur des données autres que des systèmes de particules de façon à pouvoir combiner ces outils avec d'autres modélisations.

Nous allons nous intéresser au *pipeline* complet de création artistique à l'exception

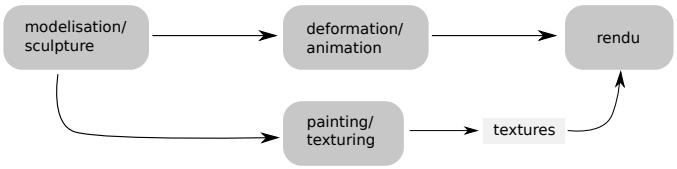


Fig. 3.24: *Pipeline* générique de création de modèle (inspiré de Burley et Lacewell [BL08]).

des problématiques de rendu à proprement parlé (fig. 3.24).

Le processus de création commence nécessairement par la création de la forme 3D. Celle-ci peut se faire en modifiant des formes existantes ou en partant de rien. La forme est alors créée par modifications locales ou globales. Généralement, le modèle est ensuite paramétré de façon à pouvoir être texturé et, en parallèle, le modèle est animé.

Ces étapes cyclent potentiellement pour améliorer la qualité de l'ensemble.

Nous allons donc voir comment réutiliser des modèles existants, comment déformer un modèle, et enfin comment le texturer. Nous ferons attention à utiliser des outils les plus intuitifs possibles en masquant au plus les difficultés techniques et, dans le cas ou ce n'est pas possible, en facilitant la compréhension des problématiques en fournissant une interface utilisateur adéquate.

Chapitre 4

Conversion en système de particules

La réutilisabilité de modèles existants, quel que soit leur paradigme de modélisation, est un aspect essentiel d'un système de création. Il est important de pouvoir étendre ou modifier un travail déjà commencé. On constate également que le processus créatif est compliqué et qu'une grande partie des utilisateurs est plus encline à partir d'un existant plutôt que de partir d'une feuille blanche. Nous allons donc voir comment convertir un modèle 3D « classique » dans notre modélisation.

4.1 Positionnement du problème

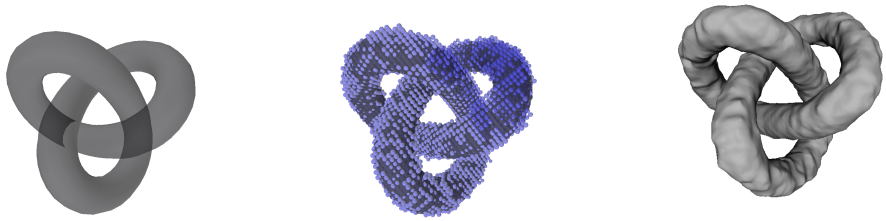
La visualisation de modèles tridimensionnels nécessite, dans le processus de rendu standard actuel, une tessellation des objets *i.e.* une collection de triangles. Les maillages triangulaires sont alors souvent utilisés comme paradigme de modélisation car ils peuvent être directement visualisés. Cette remarque nous permet de ne considérer que des maillages en entrée.

Notre objectif est de pouvoir réutiliser des surfaces tessellées dans notre modélisation *i.e.* de convertir un ensemble de triangles en volume représenté par des particules.

L'utilisation de systèmes de particules pour modéliser des formes est nouvelle, aussi, la littérature abordant ce type de problème est aujourd'hui maigre.

Dingle [Din07, Din05] propose de remplir le volume défini par la surface de façon physique, *i.e.* d'utiliser une source de matière à l'intérieur de la surface et d'ajouter des particules interagissant via des forces de type Lennard-Jones, jusqu'à obtenir un état stable de matière. Les inconvénients sont qu'ils faut préalablement déterminer les différentes composantes connexes du modèle pour que l'ensemble du volume soit récupéré, et, plus gênant, il faut que la surface soit fermée. En effet, une surface ouverte ne définit pas un volume fini et n'est donc pas modélisable ; il est alors nécessaire de chercher à fermer « au mieux » les surfaces ouvertes afin de pouvoir les considérer dans notre modélisation. Ceci nous amène à ne pas chercher une conversion exacte puisque celle-ci n'est pas toujours réalisable.

L'objectif sera d'obtenir une peau sur le système de particules qui soit visuellement proche de la surface d'origine, tout en utilisant un système de taille la plus petite possible. Dans notre modélisation, le problème devient donc celui de l'échantillonnage du volume représenté par une forme. À la lumière de notre remarque précédente, il semble intéressant de traiter les surfaces ouvertes et les surfaces fermées séparément.



(a) Modèle surfacique initial. (b) Visualisation du système de particules obtenu. (c) Peau reconstruite sur le système de particules.

FIG. 4.1: Exemple de conversion d'un modèle fermé.

Rappelons que nous utilisons uniquement des matériaux homogènes donc toutes les particules ont les mêmes propriétés de masse et de volume.

4.2 Conversion des surfaces fermées

La conversion des surfaces fermées est le cas le plus simple puisque ce type de surface défini explicitement un volume. Il s'agit alors, pour la conversion, de pouvoir définir si un point est situé à l'intérieur, ou non, du volume.

Cette problématique peut donc s'apparenter aux problématiques de voxelisation de surface, notamment utile pour effectuer des rendus par lancer de rayon. Dans un premier temps, on va chercher un ensemble de voxels intersectant la surface pour représenter le contour du volume. Comme nous l'avons vu précédemment, nous utilisons des matériaux homogènes. L'agencement des voxels est donc représenté par une structure de grille régulière plutôt qu'une structure hiérarchique. Pour déterminer le contour, l'intersection de l'ensemble des triangles avec la grille est calculée par l'algorithme présenté par Moller [AM02] reposant sur le théorème de représentation des convexes. Les cas pathologiques des voxels contenant un morceau convexe de surface sont détectés en analysant les normales des triangles contenus dans chaque voxel.

Une fois le contour déterminé, les voxels sont parcourus ligne par ligne pour déterminer les voxels intérieurs, trouvés en comptant le nombre d'intersections avec la surface : un point est situé à l'intérieur du volume si un rayon partant de l'extérieur de la surface et arrivant en ce point a subi un nombre pair d'intersections avec la surface.

Pour chaque voxel intérieur, on ajoute une particule à chacun de ses huits sommets. Les voxels sont donc de taille Δs dans chaque dimension, de façon à créer des particules tangentes (fig. 4.1).

Cette approche, que l'on peut apparenter à l'algorithme de remplissage d'un polygone convexe 2D, ne permet que la conversion d'une surface fermée.

Nous avons donc cherché une méthode dédiée aux surfaces ouvertes.

4.3 Conversion des surfaces ouvertes

Les modèles 3D disponibles ne définissent pas toujours des surfaces fermées, ni même des surfaces. Certains formats de stockage de maillage triangulaire, à l'instar

du format STL très utilisé pour la numérisation tridimensionnelle d'objets[1], perdent même la notion de surface et ne conserve que des « soupes de triangles » .

La problématique de conversion d'une surface ouverte dans notre modèle revient à chercher une fermeture « naturelle » à la surface traitée pour définir un volume. Des méthodes de nettoyage (*healing*) de surface pour fermer les trous indésirables existent (voir *e.g.* [BPK+07]). Cependant, le modèle peut avoir été volontairement modélisé comme une surface ouverte et les méthodes de *healing* ne seront généralement pas adaptées à ces cas.

À l'instar des surfaces fermées, il nous faut pouvoir déterminer si un point est situé à l'intérieur ou l'extérieur du volume M défini par sa surface ∂M. Mathématiquement, cela revient à vouloir déterminer la fonction caractéristique soit

$$\begin{array}{rccl} 1_M & : & \mathbb{R}^3 & \to & \mathbb{N} \\ & & x & \mapsto & 1_M(x) = \left\{ \begin{array}{ll} 1 & \text{si } x \in M \\ 0 & \text{sinon.} \end{array} \right. \end{array}$$

L'évaluation de cette indicatrice de volume semble délicate. Cependant, la définition même de l'indicatrice peut être utilisée comme l'ont montré Manson *et al.* [MPS08].

Décomposons l'indicatrice de volume, 1_M sur une famille $\{\phi^e_{j,k}\}$ de fonctions orthogonales constituant une base

$$1_M(\mathrm{x}) = \textstyle\sum_{j,k,e} c^e_{j,k}\, \phi^e_{j,k}(\mathrm{x}).$$

Les coefficients de la décomposition sont obtenus par produit scalaire soit

$$c^e_{j,k} = \int_{\mathbb{R}^3} 1_M(\mathrm{x}) \phi^e_{j,k}(\mathrm{x})\, d\mathrm{x} = \int_M \phi^e_{j,k}(\mathrm{x})\, d\mathrm{x},$$

du fait de la définition de l'indicatrice de volume.

La surface ∂M nous est donnée et nous cherchons le volume. Le théorème de Stokes,

$$\int_M \nabla \cdot F\, d\mathrm{x} = \int_{\mathrm{p} \in \partial M} F(\mathrm{p}) \cdot \mathrm{n}(\mathrm{p})\, d\sigma,$$

avec F une fonction continue, nous permet alors de ramener une intégrale volumique à un flux à travers la surface.

La construction d'une famille $F^e_{j,k}$ vérifiant

$$\nabla \cdot F^e_{j,k} = \phi^e_{j,k},$$

permet ainsi d'obtenir les coefficients désirés

$$c^e_{j,k} = \int_M \nabla \cdot F^e_{j,k}(\mathrm{x})\, d\mathrm{x} = \int_{\partial M} F^e_{j,k}(\mathrm{p}) \cdot \mathrm{n}(\mathrm{p})\, d\mathrm{p}.$$

Dans notre contexte, ∂M est une collection de triangles. La normale en chaque point de cette surface discrète, si elle n'est pas connue, peut être approchée au moyen d'un produit vectoriel. On note $\{v\}$ l'ensemble des sommets de la surface et n_v la

[1] La numérisation d'objet ne fournit généralement pas une surface mais bien une soupe de triangles ce qui explique l'utilisation du format STL dans ce contexte.

(a) Surface ouverte. (b) Conversion obtenue par ondelettes.

FIG. 4.2: Exemple de conversion d'une surface ouverte.

normale au sommet v. Il est possible d'estimer l'aire $A(v)$ de la cellule de Voronoï de chaque sommet au moyen de la formule des cotangentes [Lé08].

On a ainsi l'approximation

$$c^e_{j,k} \approx \sum_{\{v\}} A(v) F^e_{j,k}(v) \cdot \mathrm{n}_v.$$

Il reste alors à déterminer le choix de la base de fonctions utilisée ainsi que le calcul des fonctions $F^e_{j,k}$. Nous avons repris les choix de Manson *et al.* qui proposent l'utilisation de familles d'ondelettes, ainsi qu'une construction générique des familles $F^e_{j,k}$ associées afin d'avoir une décomposition multi-résolution et locale [MPS08].

Comme il s'agit principalement de valider l'approche, nous avons utilisé la famille d'ondelettes la plus simple, *i.e.* les ondelettes de Haar. Une fois les coefficients de la décomposition calculés, on détermine les points intérieurs au modèle en calculant la valeur de l'indicatrice et en comparant cette valeur à la moyenne de l'approximation de l'indicatrice calculée sur l'ensemble des sommets. La figure 4.2 donne un exemple de surface ouverte convertie par cette approche.

4.4 Choix d'échantillonnage

L'échelle utilisée pour représenter un modèle a évidemment un impact sur le volume que représente la forme. L'utilisateur n'est pas nécessairement conscient de cette échelle aussi on considère que le volume de l'objet est arbitraire. On utilise alors le volume d'une particule comme échelle volumique. Il nous faut alors faire une mise à l'échelle du modèle pour obtenir une conversion correcte.

Rappelons que nos objectifs sont d'obtenir un système dont la peau est proche de la surface convertie, en utilisant le moins de particules possible afin de conserver l'utilisabilité en temps réel.

La figure 4.3 présente les résultats obtenus en faisant différentes mises à l'échelle d'un même modèle. Une mise à l'échelle trop faible ne permet pas de restituer correctement la forme de départ tandis qu'une mise à l'échelle trop important implique un système de particules trop important. Dans le cas où l'utilisateur ne fournit pas le niveau de détail qu'il souhaite conserver, il nous faut mettre en place un critère afin d'obtenir un juste milieu.

Nous avons finalement fait le choix de considérer que le niveau de détail du modèle est représenté par la longueur des arêtes de la surface. On effectue une mise

à l'échelle de $\frac{1}{\bar{\ell}}$ où $\bar{\ell}$ représente la longueur moyenne d'arête. On ajoute à cela des critères sur les dimensions minimales et maximales de la boîte englobante afin de prendre également en compte le profil général de la forme.

Dans le cas de la figure 4.3, ces critères nous permettent d'obtenir la mise à l'échelle d'un facteur 84 (fig. 4.3d) et rempli donc correctement les deux objectifs de détail et de nombre de particules suffisamment faible pour une manipulation en temps réel.

4.5 Conclusion

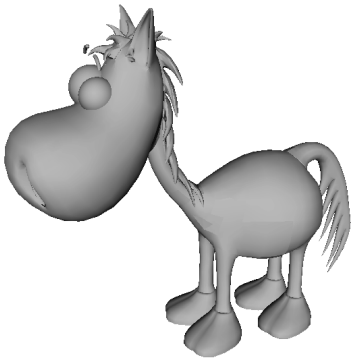
Nous avons présenté des solutions permettant la conversion de surfaces maillées dans notre modèle. Nos approches permettent la conversion de surfaces fermées et ouvertes.

La conversion en système de particules nécessite l'approximation des volumes définis par les surfaces. Dans le cas bien défini des surfaces fermées, nous avons repris le principe de remplissage des polygones bidimensionnels. Les surfaces ouvertes ne définissent pas explicitement de volume. Nous avons alors appliqué l'approximation de l'indicatrice de volume proposée par Manson *et al.* à notre problématique.

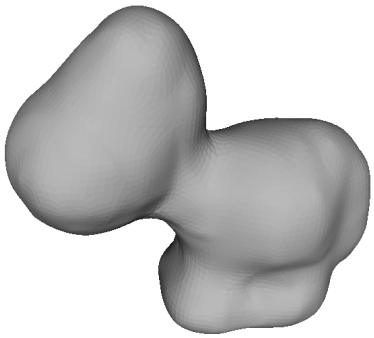
Dans le contexte des surfaces ouvertes, il serait bon d'étudier l'utilisation d'autres bases d'ondelettes pour améliorer l'évaluation de la fonction caractéristique. Tous les détails ne sont pas conservés par les approches proposées et, si celui-ci n'est pas fourni, on utilise alors une heuristique considérant la longueur moyenne comme représentative du niveau moyen de détail.

Comme nous allons le voir, les outils proposés sont également modélisés par des boules. Il serait donc également intéressant d'étudier la conversion de modèles en ensemble minimal de boules, généralement utilisée pour accélérer les détections de collisions [WZS+06, AAH+07]. Ceci permettrait, comme dans notre première approche, une première évaluation du contour des modèles et permettrait d'utiliser des modèles quelconques comme outil, en minimisant le nombre nécessaire d'éléments pour obtenir de bonnes performances.

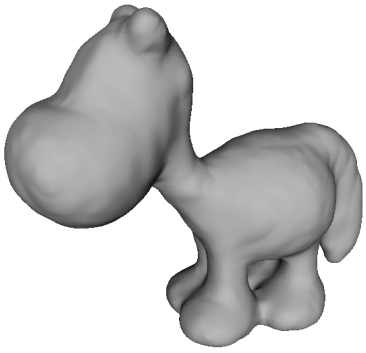
Nous allons à présent définir des outils permettant la manipulation de nos systèmes de particules, permettant la création de formes ou la modification d'un modèle converti.



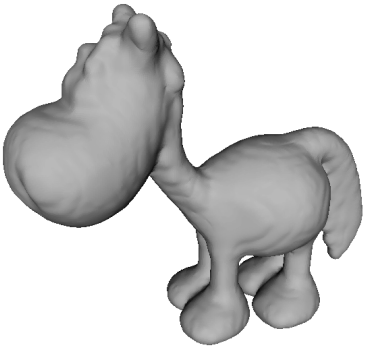
(a) Modèle original.



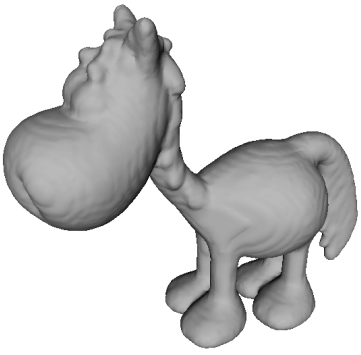
(b) Facteur d'échelle : 20 (1 167 particules)



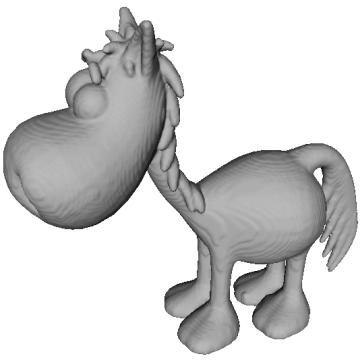
(c) Facteur d'échelle : 48 (10 742 particules)



(d) Facteur d'échelle : 84 (48 960 particules)



(e) Facteur d'échelle : 112 (112 328 particules)



(f) Facteur d'échelle : 250 (1 105 215 particules)

FIG. 4.3: Résultats obtenus pour différents échantillonnages.

Chapitre 5

Outils de déformation locale

Notre objectif initial est de recréer une expérience virtuelle de manipulation de pâte à modeler. En observant un utilisateur travaillant de la pâte à modeler, on s'aperçoit qu'un nombre restreint d'interactions est utilisé et suffit à créer la forme désirée.

Notre modélisation ne se veut pas aussi fine que celle nécessaire à la modélisation d'outils correspondants au travail que les *designers* industriels effectuent avec de la glaise (voir *e.g.* [Ber09]). Nous allons présenter cet ensemble restreint d'outils, correspondant à des opérations manuelles, que nous proposons pour manipuler la matière.

5.1 Étude des actions utilisateurs minimales

Nous allons ici nous intéresser aux interactions qu'un utilisateur en manipulant la matière avec ses doigts. Les manipulations les plus fréquemment utilisées sont :

1. ajouter de la matière ;
2. enlever de la matière ;
3. pousser la matière ;
4. pincer la matière ;
5. aplatir la matière ;
6. « boudiner » la matière.

Ces interactions sont effectuées par le bout des doigts. Pour modéliser cet outil basique, et afin de conserver une cohésion avec le reste du modèle, nous avons décidé de modéliser les outils par des boules ayant une position, un rayon, et une masse considérée infinie (p, r, ∞). La boule a, en particulier, l'avantage d'être isotrope et par conséquent d'être représentable exactement dans nos structures spatiales alignées sur les axes canoniques.

En considérant plus finement les actions précédentes, il apparaît également que l'action de boudiner la matière ou celle de l'aplatir peuvent respectivement se ramener à l'ajout de matière sous forme de sphérique *i.e.* ajouter une succession de blocs de matière sphériques, et à pousser la matière avec une surface plane qui doit alors être approchée par un ensemble de boules dans notre modélisation (à l'instar

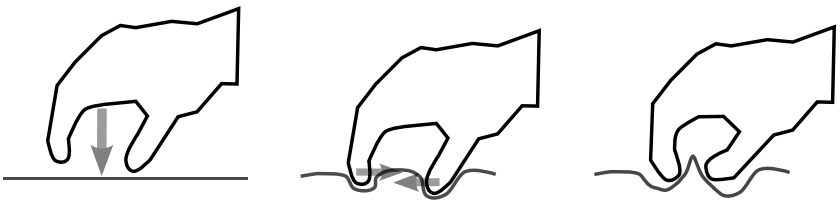


FIG. 5.1: Action physique de pincement de matière.

des méthodes de gestion de collision généralement utilisées pour les systèmes de particules [HKK07, BT07]). Il s'agit ainsi de ne pas reproduire exactement le monde réel mais de le simplifier de façon à limiter le nombre d'interactions nécessaires et le nombre d'outils proposés de sorte que l'interface soit la plus compréhensible possible.

Nous prenons également en compte la facilité d'utilisation et le fait que la plupart des interfaces homme-machine actuelles sont limitées dans le nombre d'entrées utilisateur possible. Par exemple, si on analyse certains gestes comme pincer la matière (fig. 5.1) ou enlever la matière, on observe que ce sont des gestes physiquement réalisés avec (au moins) deux doigts en deux étapes :

1. une étape de « sélection » de matière à deux doigts où l'utilisateur détermine la matière sur laquelle il va agir ;

2. une étape d'action où l'utilisateur rapproche ses doigts pour pincer la matière sélectionnée ou pour la retirer.

Une reproduction virtuelle de ces gestes serait possible mais compliquerait de façon importante l'utilisation des outils. Nous avons ainsi préféré mettre à disposition des outils fonctionnant intégralement et uniquement sur la base d'une seule entrée *i.e.* une seule boule. Nous allons à présent détailler brièvement chacune des actions minimales que nous avons déterminée.

5.1.1 Ajouter

L'ajout de matière consiste simplement à ajouter des particules de façon à remplir le volume défini par le ou les outils. Dans la mesure où plus le nombre de particules est important et plus les recherches en voisinages deviendront coûteuses, il est naturel de chercher à minimiser le nombre de particules ajoutées.
Notre outil est modélisé par une boule. Lors d'un ajout de matière, le volume de l'outil est discrétisé en volumes définis par le type de particules utilisé. Pour chaque particule de la discrétisation, on teste alors la présence de matière dans le volume correspondant pour déterminer si la particule doit effectivement être ajoutée au système (fig. 5.2).

L'ajout simultané depuis plusieurs outils ne pose pas de problème d'interférence entre les outils puisque les actions ne sont pas contradictoires. Comme la structure spatiale est mise à jour après chaque ajout de particule, l'utilisation de plusieurs outils en même temps revient à ajouter de la matière séquentiellement.

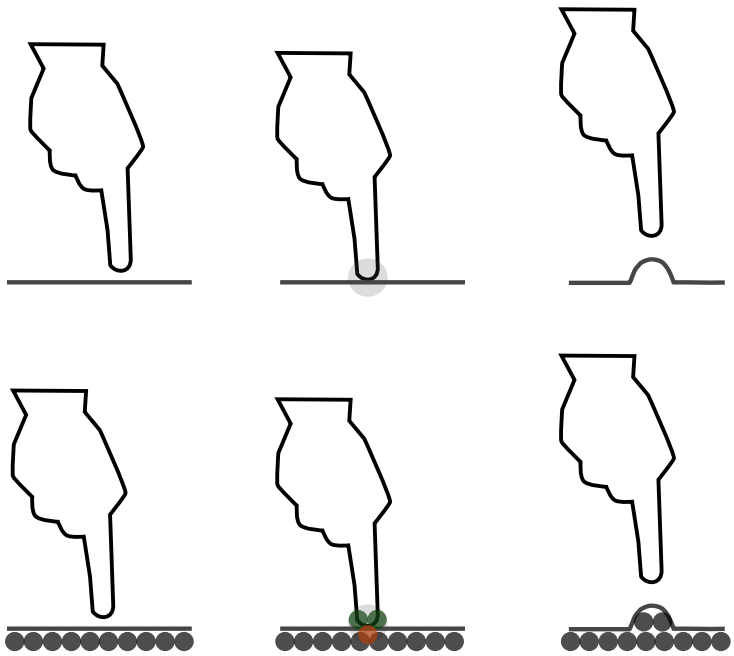


FIG. 5.2: Action de l'outil « ajouter » . En vert les particules ajoutées, en rouge une particule à l'intérieur non ajoutée au système à cause de la présence de matière.

5.1.2 Supprimer

Comme nous l'avons vu, retirer de la matière est un geste que nous simplifions dans notre modélisation. La suppression de matière est, dans notre approche, l'opération la plus directe. Lorsque l'utilisateur désire supprimer la matière située à l'intérieur de sa boule outil, une recherche des particules est effectuée et chaque particule trouvée est supprimée du système en mettant à jour les particules voisines pouvant être reliées aux particules sélectionnées par des ressorts (fig. 5.3).

L'utilisation simultanée de plusieurs outils de suppression de matière ne pose pas de problème d'interaction entre chacun des outils puisque, là encore, les actions ne peuvent pas être contradictoires. Pour chaque outil, on effectue (séquentiellement) une recherche spatiale de la taille de l'outil, on supprime les particules déterminées et on passe à l'outil suivant.

5.1.3 Pousser

Pousser la matière est une action supposée à volume constant. Aucun ajout ou suppression de matière n'est opéré ; la matière en contact avec l'outil est déplacée de façon à se trouver en dehors du volume défini par l'outil. On applique pour cela des forces radiales (fig. 5.4). Si on note p la position de l'outil, r son rayon, on applique des forces

$$F_{\text{pousser}} = \begin{cases} k_{\text{pousser}}(r + r_p - ||\text{p} - \text{v}||)\frac{\text{p}-\text{v}}{||\text{p}-\text{v}||} & \text{si} \quad ||\text{p} - \text{v}|| < r + r_p \\ 0 & \text{sinon,} \end{cases} \tag{5.1}$$

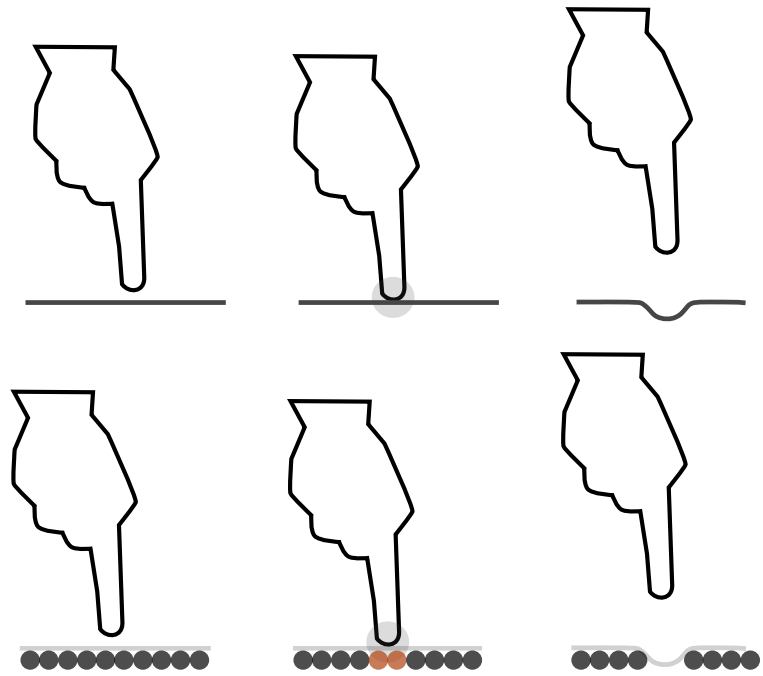


FIG. 5.3: Action de l'outil « supprimer » . En rouge, les particules situées à l'intérieur de l'outil sont supprimées.

où r_p est le rayon des particules de matière situées dans le volume de l'outil, v leur position, et k_{pousser} est une constante de force (généralement $k_{\text{pousser}} = 1$ mais un choix $k_{\text{pousser}} < 1$ permet de fournir une sensation artificielle de viscosité du matériau).

Remarquons que, dans le contexte de l'utilisation d'un matériau compressible, l'action de pousser la matière peut généralement être visuellement assimilée à celle de la supprimer. En effet, si l'outil pousse les particules à l'intérieur d'un amas de particules existantes, les particules impactées n'auront plus de contribution visible sur la surface. La question de conserver deux outils distincts est donc légitime puisque la simulation de matière incompressible n'est pas exactement garantie.

La distinction entre les deux outils est cependant légitimée dans le cadre de l'utilisation simultanée de plusieurs outils « pousser » . En effet dans ce contexte, les outils peuvent avoir des actions combinées sur le déplacement des particules et donner des résultats différents de ceux obtenus avec l'utilisation d'outils de suppression de matière.

Pour gérer l'utilisation simultanée de plusieurs outils « pousser » , on commence par déterminer, pour chaque outil, l'ensemble des outils avec lesquels il peut y avoir conflit soit l'ensemble des outils (fig. 5.5)

$$\{(\mathrm{p}', r', \infty) \text{ tq } ||r + r' + 2r_p|| > ||\mathrm{p} - \mathrm{p}'||\}.$$

Les outils sont ensuite traités séquentiellement. On commence par prédire la position de la particule après impact de l'outil courant puis on corrige la position de façon à

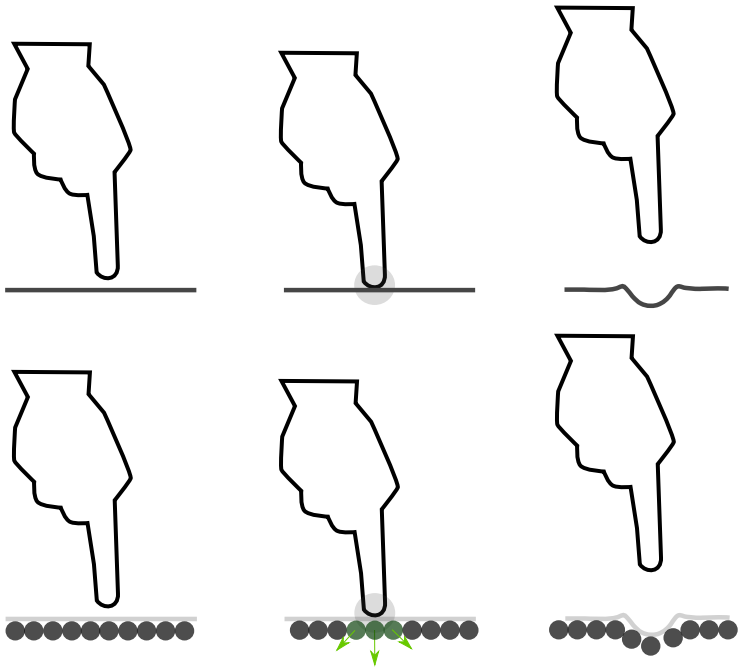


FIG. 5.4: Action de l'outil « pousser » .

FIG. 5.5: Traitement de plusieurs outils « pousser » simultanés.

respecter l'ensemble des outils en interaction et ce, tant que la position déterminée n'est pas bonne.

5.1.4 Pincer

Comme l'outil « pousser » , l'outil permettant de pincer la matière fonctionne à isoquantité de matière et tout comme l'outil « supprimer » , c'est un geste physiquement réalisé avec deux doigts. À la différence des outils précédents, l'outil « pincer » interagit avec des particules situées en dehors du volume défini par la boule outil. On cherche alors les particules situées dans un volume sphérique de rayon deux fois celui de l'outil voulu puis on déplace les particules radialement vers le centre de l'outil désiré avec un potentiel décroissant avec la distance ; on utilise un potentiel analogue au noyau *spiky* de façon à générer des forces d'intensité rapidement décroissantes (fig. 5.6). On applique ainsi les forces

$$F_{\text{pincer}} = \begin{cases} k_{\text{pincer}} \left(1 - \frac{||\text{p}-\text{v}||}{r}\right)^2 \frac{\text{v}-\text{p}}{||\text{p}-\text{v}||} & \text{si} \quad r < ||p-v|| < 2r \\ 0 & \text{sinon.} \end{cases} \tag{5.2}$$

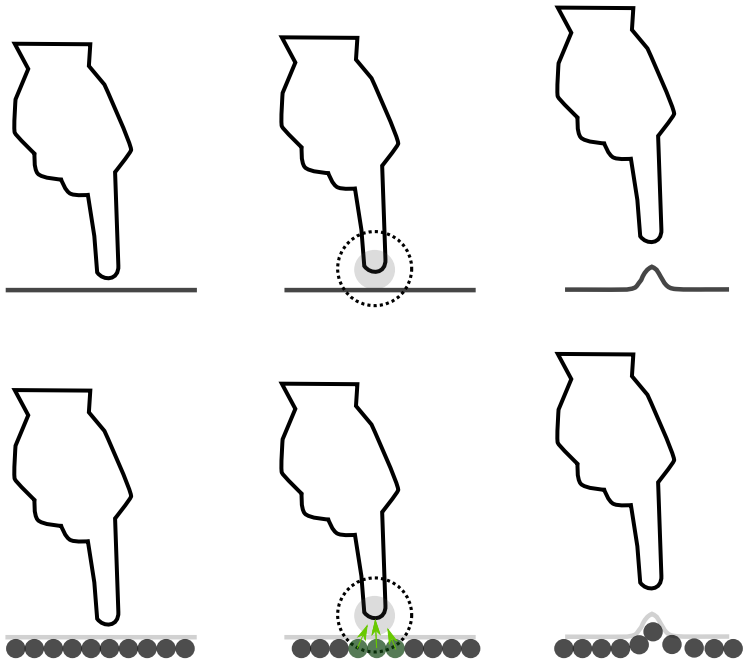


FIG. 5.6: Action de l'outil « pincer » .

L'utilisation de l'outil pincer est relativement délicate pour obtenir les résultats plaisants. Aussi, nous avons limité à un seul outil utilisable à la fois. Toutefois, il serait possible d'utiliser plusieurs outils de type « pincer » en associant aux outils en interférence potentielle, l'ensemble des particules leur étant le plus proche. De cette façon, chaque particule ne pourrait interagir qu'avec un et un seul outil.

5.1.5 Annuler/Répéter

Un des apports majeurs de la virtualisation du réel, est la possibilité d'annuler une action ou de la répéter. Ceci permet alors un processus itératif d'essais-erreurs jusqu'à ce que la forme voulue soit modélisée ce qui serait difficilement réalisable avec un matériau et une oeuvre physique.

Le coeur de notre modélisation repose sur les particules et la surface reconstruite repose sur la densité des particules. Dans la mesure où on considère l'utilisation de particules ayant les mêmes propriétés physiques, cette densité est entièrement déterminée par la position des particules.

Si on analyse les types d'interaction possibles on se rend compte qu'ils sont au nombre de trois :

1. **création de matière** via l'outil d'ajout ;
2. **suppression de matière** via l'outil de suppression ;
3. **déplacement de matière** via les outils pousser, pincer ou sous l'effet de la simulation.

Il est évident que l'action de création est l'opposé de l'action de suppression. L'opposé d'un déplacement est le déplacement opposé. Comme notre modèle re-

pose fondamentalement sur les positions des particules, il suffit alors, pour chaque particule modifiée, de stocker un vecteur représentant

la position de la particule pour une création ou une suppression de matière,

le déplacement appliqué dans le cas d'une simulation ou de l'utilisation des outil pousser ou pincer,

et de connaître le type d'action pour donner la bonne sémantique au vecteur. Lorsqu'une action est annulée, l'action « inverse » est ajoutée à la liste des actions répétables. Inversement, si l'utilisateur annule la répétition, l'action est à nouveau insérée dans la liste des actions annulables. Le système de vases communiquants est arrêté lorsqu'une nouvelle action est enregistrée.

Enfin, pour gérer une annulation sur plusieurs niveaux, il est nécessaire d'ajouter un identifiant aux particules. En effet, dans une succession d'action, une particule peut être déplacée puis supprimée. Comme nous venons de voir, l'opération inverse consiste à créer la particule et à la déplacer. Dans ce cas, il nous faut être en mesure d'associer la translation sur la particule nouvellement créée.

Afin de limiter le nombre de niveaux et comme un appel unique à un outil peu avoir un impact très restreint, on enregistre les actions par geste. On considère qu'un geste commence lorsque l'utilisateur active l'outil et se termine lorsque l'outil est relâché.

Ainsi, notre mécanisme d'annulation/répétition est globalement peu coûteux et permet de gérer un nombre quelconque de niveaux d'annulation/rétablissement. Comme il est basé sur les positions des particules, il permet également la mise à jour incrémentale[1] de la surface.

5.1.6 Outils de sélection

Une autre possibilité offerte par le virtuel est la possibilité de figer une partie d'une oeuvre tout en permettant la modification du reste de l'oeuvre. Ceci permet d'améliorer une sous-partie d'un modèle sans perturber les autres sous-parties et en gardant une vue d'ensemble du modèle final. Si le travail « par morceaux » est réalisable dans le monde réel, le découpage des parties éditables est contraint par la physique du monde et l'outil informatique permet de s'affranchir de ces contraintes physiques pour donner une plus grande liberté à la création.

Il est donc intéressant de proposer des outils de sélection de volume. L'outil le plus basique est évidemment l'outil de sélection par « trappe » . L'utilisateur définit un volume, typiquement de forme sphérique ou parallélépipédique, et le système détermine les particules situées à l'intérieur via une recherche spatiale. Ce type de sélection est donc immédiat avec les outils déjà présentés mais l'obtention d'une sélection complexe sur la seule base de boules et de parallélépipèdes (alignés sur les axes canoniques) devient délicate. Nous avons donc mis en place deux autres types de sélections plus évoluées que nous allons présenter.

5.1.6.1 Sélection par composante connexe

Notre modélisation, si elle permet une création libre, n'explicite pas le fait que l'on puisse créer plusieurs composantes connexes par exemple. Deux objets visuel-

[1] dans le cas où l'action est de type déplacement, la fonction implicite — et donc la surface — est mise à jour dans le voisinage de la particule avant et après déplacement

lement distincts peuvent être créés par un unique système de particules. Or il peut être intéressant de sélectionner chacun des objets individuellement.

Deux approches sont envisageables pour cette problématique. La première consiste à distinguer les deux composantes connexes surfaciques (par exemple en étudiant le graphe de voisinage des triangles), puis, pour la surface sélectionnée, d'utiliser une représentation volumique permettant de tester quelles particules sont situées à l'intérieur du volume déterminé par la surface. Cette méthode serait relativement coûteuse et, dans la mesure où l'habillage reconstruit est situé légèrement à l'intérieur du volume des particules, la sélection pourrait être peu précise.

La seconde consiste à travailler uniquement sur la base du système de particules. La surface reconstruite est basée sur la densité de particules qui dépend du voisinage de particules. Pour déterminer l'ensemble des particules générant un morceau de surface, on peut ainsi procéder de façon gloutonne. On effectue une recherche de voisinage autour du point déterminé. Puis, tant que le voisinage n'est pas vide, pour chaque particule trouvée précédemment, on détermine le particules voisines qui n'ont pas encore été sélectionnées (fig. 5.7).

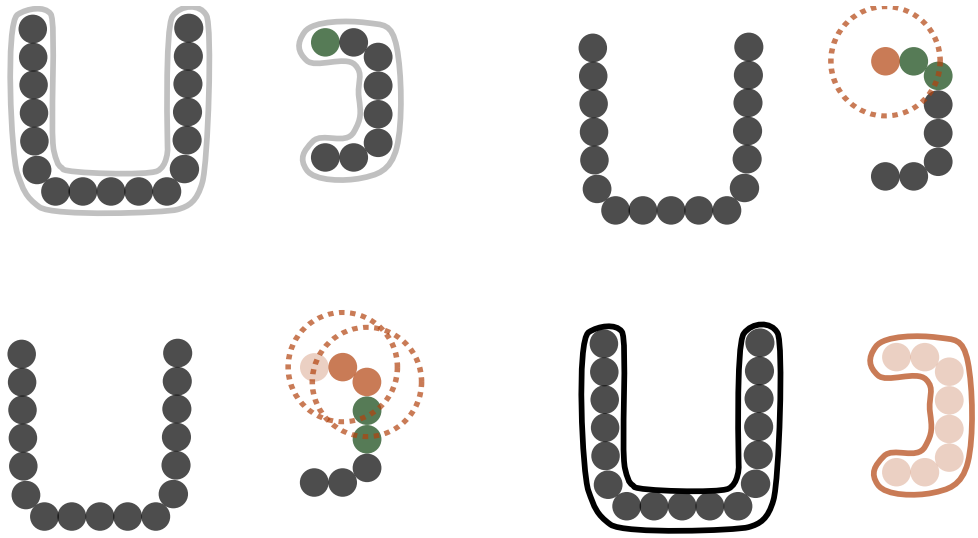


FIG. 5.7: Sélection gloutonne d'une composante connexe de particules.

Cet outil de sélection permet de retrouver l'ensemble des particules générant une surface close dans le système. Il est donc très pratique pour découper un système en sous-systèmes disjoints.

Cependant, l'outil ne permet pas d'extraire de l'information à partir d'un système connexe (au sens du voisinage sur les particules) de particules. Nous allons tenter d'extraire plus de sens de nos systèmes de particules.

5.1.6.2 Sélection par segmentation

Extraire automatiquement les différentes parties « significatives » d'un maillage, *i.e.* segmenter une surface, est un problème complexe. La surface ne porte pas explicitement la sémantique qu'un humain pourra lui associer. L'absence des informations liées à la topologie de la surface ajoute un niveau de complexité.

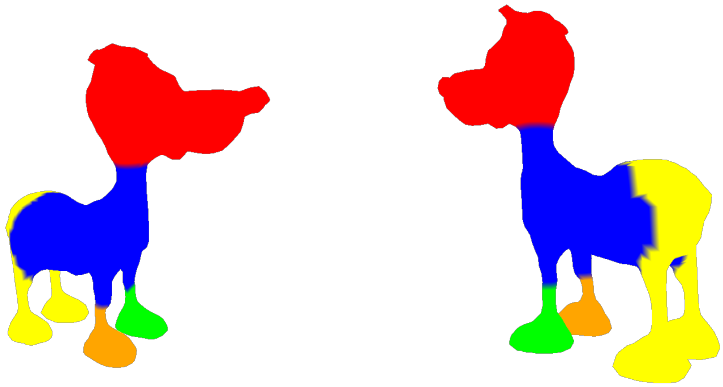


FIG. 5.8: Segmentation de nuage de points surfacique. Seuls les sommets de la surface ont été utilisés et un rendu avec une couleur par sommet a été réalisé.

Richtsfeld et Vincze [RV09] proposent une solution basée sur une réflexion radiale. L'idée repose sur le fait que beaucoup de modèles possèdent une partie centrale sur laquelle sont rattachées des parties ayant séparément du sens. Leur approche consiste ainsi à déterminer un centre du nuage de points et un rayon. Les points du nuage subissent une première transformation éloignant les points les plus proches du centre et déplaçant peu les points plus éloignés. L'enveloppe convexe du nuage ainsi transformé est calculée puis est modifiée pour que les points voisins de l'enveloppe (donc des points initialement proche du centre du modèle) se retrouvent à l'extérieur de l'enveloppe et que cet ensemble de points fournisse le corps central du modèle. Les autres parties sont alors les différentes composantes connexes du nuage initial auquel on a retiré la partie centrale.

Cette approche, basée sur une observation des modèles usuels, est présentée pour des nuages de points supposés proches de la surface. Néanmoins, comme l'approche repose principalement sur l'utilisation de l'intérieur d'une enveloppe convexe, donc d'un volume, l'approche peut ainsi s'appliquer directement sur des nuages de points volumiques.

Les résultats sont très convaincants pour des modèles faisant apparaître un corps central naturel (voir fig. 5.9). On peut également noter la similitude, en nombre de segments et dans la taille de ceux-ci, entre le résultat obtenu sur le nuage volumique (fig. 5.9) et sur le nuage surfacique (fig. 5.8), validant ainsi l'utilisabilité de l'approche dans un contexte volumique comme le notre.

L'approche est cependant moins robuste sur des modèles ne présentant pas de corps central très distinct (fig. 5.10). Le problème vient en partie du calcul initial du centre du modèle pour lequel nous avons utilisé le barycentre du système et non le centre de la sphère minimale comme proposé par Richtsfeld et Vincze. Une correction manuelle peut fournir les résultats attendus intuitivement.

Ce choix du centre rend délicate la création d'un outil entièrement automatique et fonctionnant sur la plupart des modèles. Cependant l'outil présenté permet déjà une première sélection généralement naturelle. Un travail sur l'ergonomie de cet outil de sélection serait à effectuer afin d'avoir des indications de l'utilisateur sur le découpage à effectuer.

Ce genre d'outils de découpe est très intéressant dans un contexte grand public car il permet une réutilisation très aisée de modèles existants. L'utilisateur pourrait

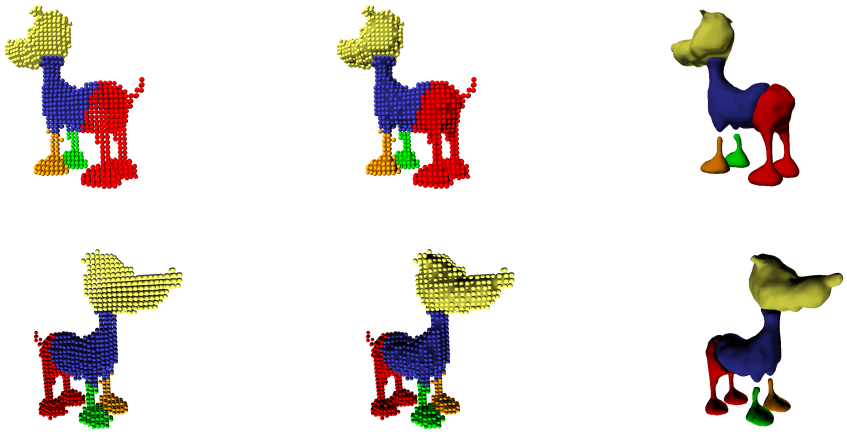


FIG. 5.9: Segmentation de nuage de points volumique. Les différents segments trouvés sont représentés par des couleurs différentes ; les segments volumiques et leur surface associée sont superposés et rendu séparément.

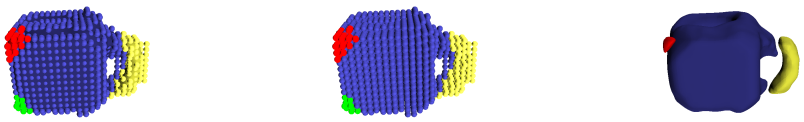


FIG. 5.10: Segmentation sur un modèle volumique peu symétrique.

découper des morceaux d'intérêts sur différents modèles puis les assembler pour créer un modèle original permettant ainsi la réutilisation et la création de modèles, de façon comparable à ce qui est proposé par Spore [HRE+08] ou Meshmixer [SS10].

5.2 Prototypes et interface utilisateur

Pour démontrer et tester l'utilisabilité de notre modèle, nous avons été amenés à réaliser plusieurs prototypes au cours de nos travaux. Nous avons, pour cela, codé un ensemble d'outils C++ séparés en trois catégories fournissant des briques logicielles de base :

maths ensemble d'outils bas niveau, permettant la manipulation de vecteurs, matrices et structures de recherches spatiales. Les structures sont templatées de façon à permettre une utilisation la plus générique possible ;

modeler ensemble d'outils géométriques reposant sur des composants de *maths*, permettant la manipulation de maillage, l'extraction de surface implicite, et d'autres algorithmes de traitement de surface ;

clay ensemble d'outils permettant la manipulation de systèmes de particules reposant sur les composants *maths* et *modeler*. Les objets de ce composant gèrent la simulation physique et la reconstruction de surface. Ce sont les seuls objets

exposés à l'extérieur de notre environnement. Ce sont les objets utilisés pour la création des expériences utilisateur.

Tout le code est écrit en C++ standard de façon à compiler sur le plus grand nombre de plateformes. Pour l'interface utilisateur, nous avons implenté des *plugins* permettant d'interfacer notre modeleur avec 3DVIA Virtools [Dasd] et 3DVIA Studio [Dasc] pour la création d'expériences virtuelles interactives. Ces plateformes ont été utilisées pour leurs moteurs de rendu multiplateformes performants, et pour la création de l'interface utilisateur. 3DVIA Virtools dispose également du VR Pack, une bibliothèque pour la gestion des périphériques de réalité augmentée standards à laquelle nous avons fait quelques ajouts.

L'architecture globale est synthétisée en figure 5.11.

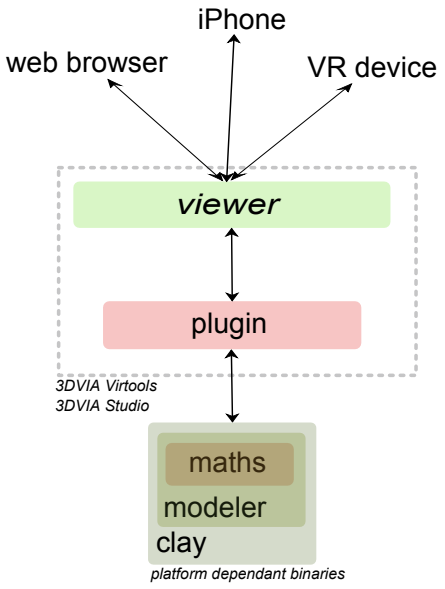


FIG. 5.11: Architecture générale des prototypes.

5.2.1 Prototype *online*

L'expérience utilisateur d'un logiciel passe par plusieurs stades dont le premier est l'installation qui doit être la plus simple possible. L'utilisation de 3DVIA Studio est appropriée à ce scénario puisqu'il s'agit d'un lecteur d'expérience donc ne nécessite, pour une utilisation standard, que l'installation du logiciel lui même. L'expérience, au sens logiciel, est ensuite téléchargée de façon transparente et jouée sur la machine client. Dans notre cas, nous devons également fournir un installeur sommaire, déployant le *plugin* 3DVIA Studio et les binaires du modeleur, dont l'installation peut se faire en un clique souris.

De plus, la mouvance actuelle est d'intégrer le plus possible le réseau dans l'expérience afin de enrichir celle-ci. Nous avons donc développé un premier proto-

type utilisant le moteur de rendu de 3DVIA Studio intégré aux navigateurs web afin de pouvoir réutiliser au mieux les modèles déjà existants sur les bases 3DVIA.com. Ceci permet également la sauvegarde des modèles créés sur ces mêmes bases.

L'interface utilisateur du logiciel devient ensuite le point clé de l'expérience et doit être *intuitive*, *i.e.* comprise spontanément, et *naturelle*, *i.e.* conforme au bon sens. Notre choix d'outils de modifications locales a été réalisé à cette fin. Il nous reste à déterminer comment l'utilisateur va pouvoir contrôler les outils pour interagir avec la matière.

La plupart des outils de modélisation 3D actuels sont basés sur des représentations purement surfaciques. Dès lors, l'utilisation d'un outil modifiant la surface est contrainte d'avoir lieu *sur* la surface. Pour localiser l'entrée utilisateur (généralement une position de souris), on utilise alors du *picking* permettant de faire un lien entre le curseur souris et un point sur la surface. Un rayon est généralement lancé depuis la souris et traverse la scène jusqu'à trouver un élément de la surface.

La littérature s'est intéressée depuis longtemps à l'utilisation de périphériques haptiques permettant d'augmenter encore le réalisme en fournissant un retour de force [GH91, PKC08]. Cependant, ce type de matériel reste onéreux et est très peu utilisé du grand public. Nous avons donc opté pour un contrôle au moyen de périphériques classiques de type clavier/souris.

Dans notre cas, la modélisation est entièrement volumique et il n'y a donc pas de réelle raison à vouloir limiter l'interaction avec la matière de se faire au niveau de la peau du système. La problématique devient alors celle du contrôle d'un objet 3D non contraint au moyen d'un périphérique 2D, où le contrôle est à la fois relatif à la capacité à déplacer l'outil mais aussi à la maîtrise de son placement.

Dans un premier temps, nous avons voulu étendre l'utilisation du *picking* 2D à un déplacement tridimensionnel. Nous avons pour cela ajouter un plan infini à la scène et utilisé ce plan comme moyen de contrôle ; le plan est alors la seule surface considérée lors du *picking* (fig. 5.12). L'outil est ainsi positionné à l'endroit de l'intersection du rayon issu de la souris et du plan infini.

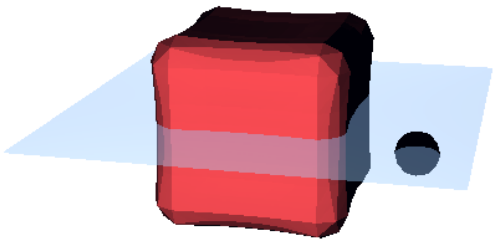


FIG. 5.12: Contrôle de l'outil par plan.

L'avantage de la solution est le fait que ce type d'approche ne repose pas sur la nature des objets utilisés. De plus, l'intersection rayon-plan est très peu coûteuse alors que l'utilisation de *picking* nécessite généralement la construction d'une structure arborescente pour accélérer la détection de l'intersection rayon-surface. Le problème devient alors le contrôle intuitif du positionnement d'un plan dans l'espace, problème qui est assez délicat, et également de la surcharge visuelle entraînée par la présence d'un élément tierce dans la scène.

Nous avons alors opté pour un contrôle non plus indirect *i.e.* un positionnement dépendant de l'entrée utilisateur *et* d'un autre élément de la scène — que celui-ci soit la surface manipulée elle même ou, comme présenté précédemment, un autre objet comme le contrôle proposé au moyen d'un plan — mais direct de l'outil. Pour se faire, nous avons donc considéré que la caméra utilisée dans la scène modélise la vue de l'utilisateur. L'outil, supposé être la main de l'utilisateur, se déplace alors relativement à cette vue. Les déplacements en (x, y) de la souris sont reproduits dans le repère caméra. La profondeur est contrôlée par deux touches clavier équivalentes à « plus » ou « moins » de profondeur ou au moyen d'un contrôleur spécifique (fig. 5.13) et opère, naturellement, dans le sens de la profondeur de la vision.

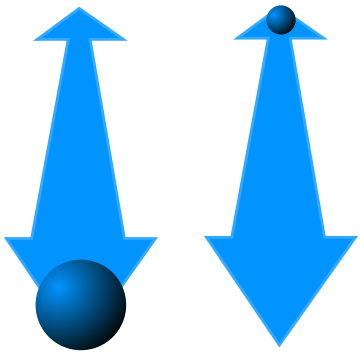


FIG. 5.13: Contrôleur de profondeur par *slider*. La taille de l'icône symbolisant l'outil est ajustée en fonction de la profondeur.

Cette approche conserve l'agnosticisme vis à vis du type de données traitées sans surcharger la scène visuellement tout en permettant un contrôle naturel.

Il reste alors le problème primordial de la maîtrise du positionnement de l'outil par rapport à la matière existante. Même si les écrans (auto)stéréoscopiques se développent, l'utilisation d'écrans purement 2D reste le matériel commun. Le positionnement relatif de l'outil par rapport à l'outil peut alors être problématique (fig. 5.14). Il faut pouvoir positionner relativement le « couple » matière-outil, en plus du besoin de positionnement de l'outil par rapport au point de vue.

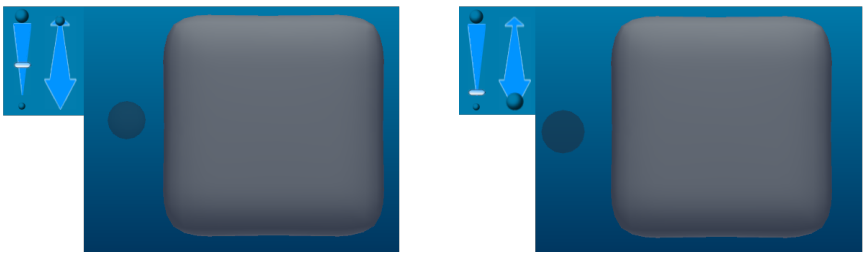
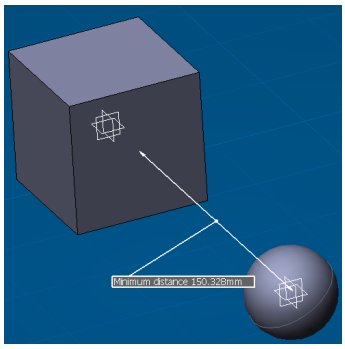


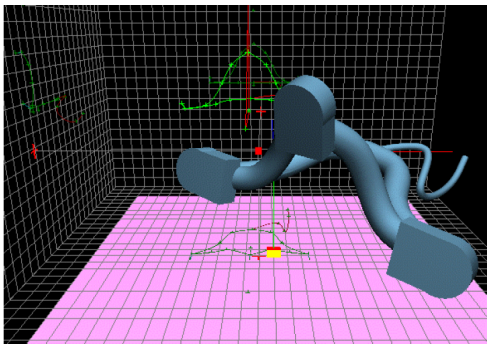
FIG. 5.14: Problème de détermination de la position relative de l'outil et de l'objet.

Cette problématique peut être formulée comme celle de l'évaluation d'une distance entre deux objets d'une scène 3D visualisée en 2D. Certaines solutions répondent à la problématique en ajoutant de l'information visuellement dense (fig. 5.15a)

ou cognitivement complexe (fig. 5.15b).



(a) Visualisation au moyen d'une flèche et de texte dans le logiciel CATIA.



(b) Visualisation au moyen de plans de projection [LG00].

FIG. 5.15: Solutions actuelles de visualisation de distances entre des objets d'une scène 3D.

Nous proposons une solution permettant une visualisation intuitive, intégrée à la scène et implémentable à faible coût. Notre idée consiste simplement à utiliser le fait que lorsqu'on approche une lumière d'un mur, le halo de lumière sur le mur se rétrécie et l'intensité lumineuse dépend de la distance des points du mur à la source de lumière. Nous retirons de l'information intuitivement en observant la tâche de lumière et l'intensité des points de la tâche. Il est donc possible de visualiser un champ de distance en utilisant une source de lumière. Si l'outil a une direction privilégiée, on utilisera une source de lumière directionnelle, autrement une source omnidirectionnelle. Les lumières disposent également, en général, de paramètres permettant de limiter leur action dans l'espace de façon à avoir une information locale.

Cette solution a fait l'objet d'un dépôt de brevet, en cours de validation, d'aide à la décision par modification de rendu.

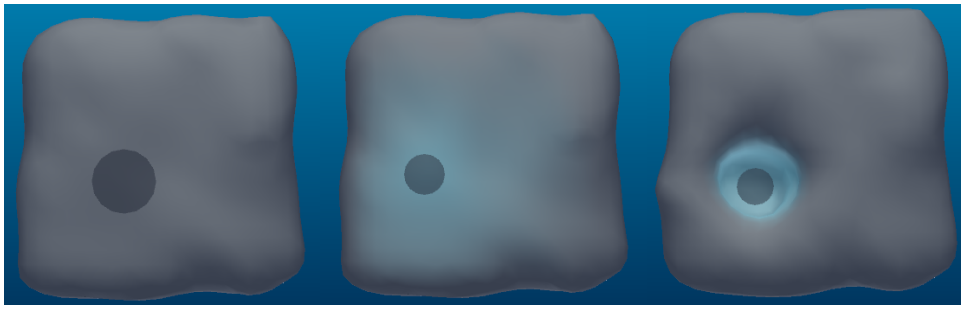


FIG. 5.16: Visualisation de la distance relative entre deux objets par une source de lumière. De gauche à droite : l'objet n'est pas dans la zone d'influence de l'outil. L'objet entre dans la zone d'influence de l'outil. L'outil est à l'intérieur de l'objet.

Nous avons ainsi mis en place un prototype accessible, à la fois sur le plan

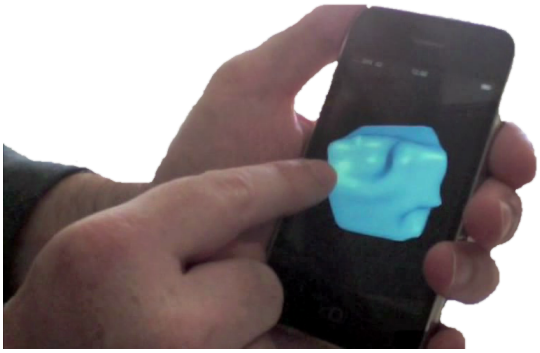


FIG. 5.17: Prototype iPhone.

matériel, car il nécessite peu d'intervention utilisateur, et le plan *logiciel* grâce à une solution intuitive de manipulation spatiale de l'outil au moyen de périphériques standards. Si ce prototype est globalement satisfaisant, l'expérience utilisateur peut encore être améliorée en fournissant des interactions plus proches de celles du monde réel. Nous allons présenter brièvement d'autres prototypes, reposant sur l'utilisation de matériel spécifique, mis en place dans ce but.

5.2.2 Vers une interface tangible

La souris permet une manipulation précise mais est peu fidèle des mouvements effectués dans le monde réel. L'iPhone a grandement contribué à l'émergence des technologies tactiles que l'on trouve aujourd'hui dans de plus en plus d'appareils, principalement portables et tablettes. Les technologies tactiles sont attrayantes pour notre modélisation car elles permettent d'utiliser directement le doigt comme outil et ainsi de reproduire un contact *direct* avec la matière. Nous avons donc développé un prototype tournant sur plateforme iPhone.

Il y a cependant deux inconvénients à ce type d'appareil. Le premier est lié à la puissance de calcul qui permet difficilement d'obtenir des résultats fluides en temps réel. De plus, le gain apporté par la manipulation au doigt est à nuancer par la difficulté du contrôle de la profondeur de l'outil sous le doigt. Utiliser plusieurs doigts à cette fin est délicat du fait de la taille de l'appareil. Nous avons alors opté pour une solution basée sur un lancer de rayon. Dans le cas où il n'y a pas de matière, on positionne l'outil à une profondeur calculée en fonction de la boîte englobante de la matière.

Ce prototype n'est, au final, que peu utilisable. Cependant, ce type d'interface homme-machine, qu'on peut qualifier de naturelle est grandement souhaitable, pour peu qu'elle fournisse un contrôle aisé en trois dimensions et non seulement deux.

Nous avons alors testé divers matériels de réalité immersive. Leur utilisation n'est, à l'heure actuelle, pas destinée au grand public, mais le deviendra probablement dans un futur relativement proche. Nous avons ainsi développé un prototype pour une station de travail de type *Personal Space Station* (fig. 5.18) et un environnement de type *cave* (fig. 5.21).

L'intérêt de ce type de matériel est qu'un objet peut être suivi selon ses 6 degrés de libertés (trois degrés en translation et trois degrés en rotation) ce qui permet une interaction tridimensionnelle fort souhaitable dans une modélisation volumique

FIG. 5.18: Une *Personal Space Station*.

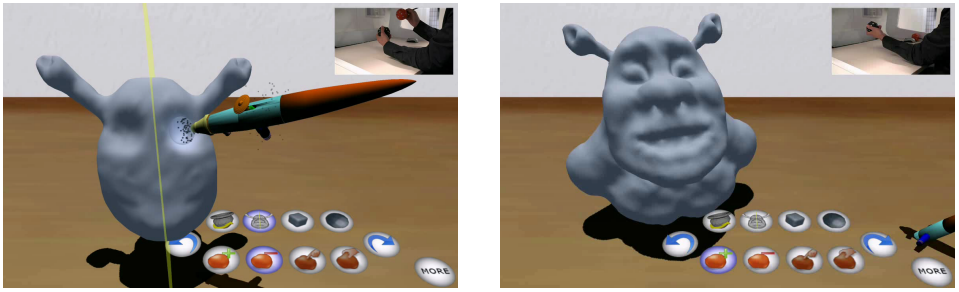


FIG. 5.19: Interface réalisée et exemple de modèle produit par un utilisateur novice.

comme la notre. Les sensations créées par nos prototypes sont très prometteuses. L'utilisation devient plus ludique et plus aisée, même pour un utilisateur n'ayant pas de connaissance préalables en modélisation 3D (fig. 5.19).

Des nouvelles approches de modélisation sont également possibles. Nous avons par exemple pu imiter un tour de potier en faisant tourner la matière selon un axe de rotation. Le résultat produit est équivalent à celui qu'on pourrait obtenir avec des surfaces de révolutions. Néanmoins, les manipulations faîtes par l'utilisateur ont alors une signification physique et rendent l'outil plus facilement utilisable.

Le *cave* utilisé présente un environnement plus immersif. L'utilisateur est entouré de 3 écrans, deux en angle et un au sol, lui permettant d'immerger son corps intégralement. L'avantage est qu'on peut alors modéliser des objets à l'échelle humaine, tout en projetant son corps dans l'univers créé. Il y a ainsi une complémentarité avec l'utilisation du PSS qui permet de créer des objets plus petits avec une immersion moins importante. Le couplage de ces deux matériels est donc très intéressante pour la création d'un monde virtuel détaillé.

5.3 Conclusion

Les prototypes réalisés sont prometteurs. L'aspect pâte à modeler rend le logiciel plus attrayant car moins formel. De plus peu d'outils, à la signification intuitive, suffisent pour créer des modèles facilement. L'utilisation d'outils de sélection puissants permettant, entre autres, de « copier-coller » de la matière pour réutiliser des modèles existants est également intéressante.

Les matériels immersifs, de type PSS ou *cave*, permettent une expérience plus réaliste. L'utilisation de gants, comme ceux présentés par Wang et Popović [WP09] permettraient une interaction a plusieurs doigts encore plus naturelle. Il serait intéressant de chercher à reproduire des sensations tactiles sans utiliser de matériel

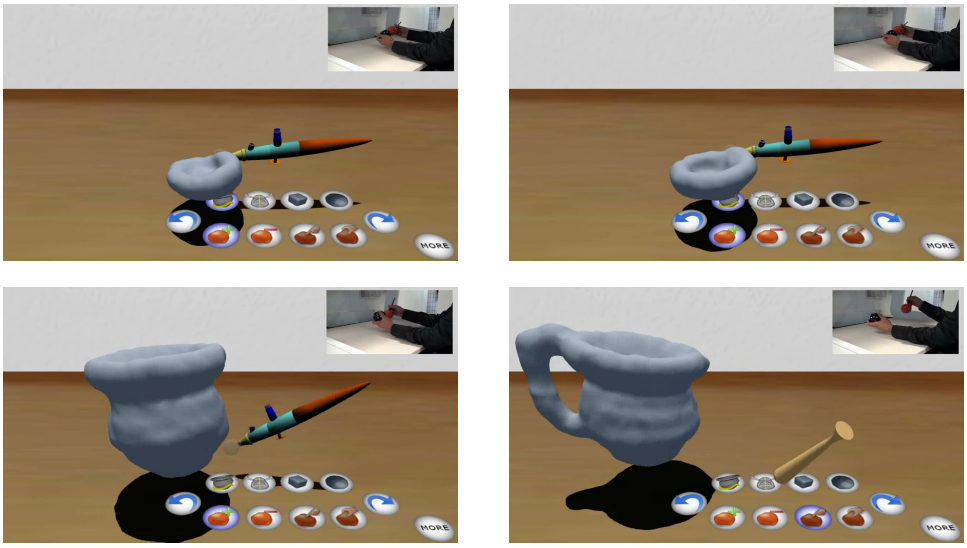


FIG. 5.20: Mode potier.

FIG. 5.21: Prototype un environnement de type *cave*.

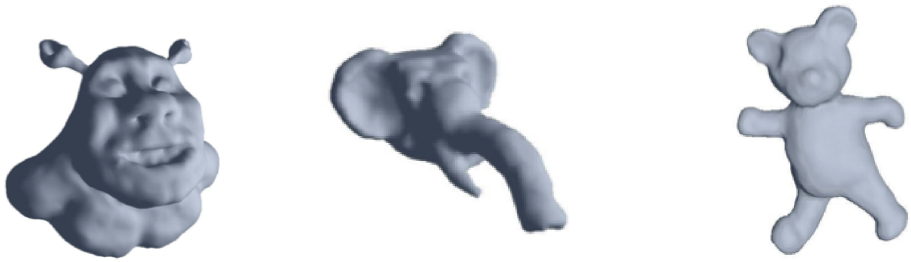


FIG. 5.22: Quelques réalisations obtenues lors de tests utilisateur.

haptique très onéreux. Ces sensations *pseudo-haptiques* devraient ainsi reposer sur les autres sens à disposition, que ce soit la vue ou l'ouïe, à l'instar de notre solution de perception de distance au moyen d'une lumière.

On peut ainsi envisager la création de nouveaux paradigmes de création 3D ; un couplage de ces prototypes avec des outils de reconstruction 3D à partir de vidéo permettrait d'intégrer le réel au virtuel et de l'utiliser, comme outil ou comme décor, pour enrichir l'expérience.

Chapitre 6

Outils de déformation globale

Les outils présentés dans la section précédente ne permettent que des modifications locales au sens où le matériau n'est modifié qu'à proximité de l'outil utilisé. La création d'un modèle nécessite souvent un processus itératif d'essai-erreur, dans lequel une esquisse peut créer une nouvelle idée d'une forme proche de la forme actuelle mais nécessitant une transformation plus globale. Par exemple, on peut vouloir modifier la courbure d'une anse ou la position du bras d'un personnage. L'utilisation des outils précédents serait alors fastidieuse. De telles opérations nécessitent une prise en compte d'un voisinage plus important que celui utilisé précédemment, voire la prise en compte de l'ensemble des particules du système.

Notre objectif est le développement d'un outil ne limitant pas la créativité de l'utilisateur. Devoir décomposer une déformation importante en un ensemble de déformations locales serait un frein à la création. Il semble donc intéressant de compléter les outils précédents avec une approche de déformation globale. L'utilisateur doit pouvoir définir la partie de son modèle qu'il souhaite déformer et imposer des contraintes pour obtenir la forme désirée. Ce processus de modification de forme est supposé réalisé à volume (quasi) constant.

Par ailleurs, la déformation implique l'existence d'une forme de départ. Nous avons vu qu'il était possible de transformer un modèle surfacique polygonal en système de particules. On pourra, par exemple, déformer un modèle existant, avant de le transformer en système de particules pour pouvoir changer la topologie de la forme. L'outil proposé doit donc pouvoir être générique pour rendre possible une utilisation en amont de notre modèle particulaire tout en permettant également la manipulation de celui-ci.

Enfin, notre créativité est, en règle générale, guidée, si ce n'est soumise, par les règles physiques du monde qui nous entoure. Les déformations produites devront donc être physiquement plausibles pour sembler naturelles.

Nous allons dresser un état de l'art rapide des méthodes répondant à la problématique de déformation globale. Après une analyse des méthodes existantes pouvant répondre à nos besoins, nous présenterons la méthode existante que nous avons retenue ainsi que les ajouts que nous lui avons apportée.

6.1 État de l'art

La littérature abordant les problématiques de déformation de modèles 3D est très riche. Une des premières approches, proposée par Sederberg et Parry [SP86], étend le

principe de manipulation des surfaces de type *spline* ; ces surfaces polynomiales sont définies par un maillage de contrôle déformable. Pour reprendre l'idée du maillage de contrôle, les auteurs proposent de plonger la surface dans une grille volumique de contrôle dont on peut manipuler les sommets indépendamment les uns des autres. La surface est déformée en repérant les sommets de la surface initiale dans la grille puis en interpolant leur nouvelle position par rapport aux sommets de la grille de contrôle déformée. Sedeberg et Parry proposent d'utiliser la famille de polynômes de Bernstein pour calculer les nouvelles positions. En contrôlant la déformation de chaque voxel de la grille de contrôle, il est également possible d'assurer conservation du volume lors de la déformation.

Coquillart [Coq90] propose une extension à l'approche précédente en autorisant la grille de contrôle à contenir non plus seulement des parallélépipèdes mais également des prismes. Il devient ainsi possible de créer une grille de contrôle qui capture mieux la forme initiale de l'objet et permet des déformations plus naturelles de la surface.

Si ces approches sont puissantes et permettent de contrôler de façon locale et globale une surface, elles peuvent néanmoins paraître peu intuitives pour l'utilisateur qui doit manipuler une grille de contrôle et non directement la surface. Hsu *et al.* [HHK92] proposent de lever ce problème en masquant intégralement la grille de contrôle. L'idée est de ne plus contrôler la grille de contrôle mais de déduire celle-ci à partir de contraintes. L'utilisateur détermine des contraintes *i.e.* les déformations à appliquer à certains points de la surface puis le système détermine une configuration de grille de contrôle correspondant à ces contraintes. L'approche par moindres carrés permet de gérer les cas sous ou sur contraints.

Afin de reproduire des déformations physiquement plus réalistes, certaines approches ont repris des formulations de type éléments finis, soit surfaciques [GHDS03], soit volumiques [MDM+02], au prix d'un coût de résolution généralement incompatible avec le temps réel. Elles ne sont par ailleurs pas toujours souhaitables puisque même si l'utilisateur veut obtenir des formes plausibles, il ne veut pas nécessairement contraindre sa création virtuelle à être physique. Néanmoins, les surfaces de haute qualité obéissent généralement au principe de minimisation des variations des courbures. Plusieurs approches reposent sur des approximations des énergies de plaques minces. Botsch et Kobbelt [BK04] dérivent le minimum de ces énergies en écrivant le problème sous forme variationnelle. Ils se ramènent ainsi à un problème de laplacien avec conditions au bord et contrôle de la continuité.

Un ensemble d'approches pour la déformation de maillage repose sur les coordonnées différentielles plutôt que sur les coordonnées spatiales. Si on note

$$L(\mathrm{v}_i) = \mathrm{v}_i - \frac{1}{\#\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} \mathrm{v}_j$$

un opérateur de coordonnées différentielles du sommet v_i, on a

$$L(T(\mathrm{v}_i)) = (\mathrm{v}_i + \mathrm{t}) - \frac{1}{\#\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} (\mathrm{v}_j + \mathrm{t}) = L(\mathrm{v}_i)$$

et

$$L(R(\mathrm{v}_i)) = R\mathrm{v}_i - \frac{1}{\#\mathcal{N}_i} \sum_{j \in \mathcal{N}_i} R\mathrm{v}_j = R(L(\mathrm{v}_i)).$$

L'opérateur de coordonnées différentielles est invariant par translation et commute avec une rotation. En faisant l'hypothèse que la propriété de commutativité, valable

globalement, peut se transposer localement du fait des variations lisses des plans tangents, Lipman *et al.* [LSCO+04] proposent une déformation en deux étapes. Dans une première passe, des transformations sont appliquées au maillage. Des rotations locales à chaque sommet sont estimées et appliquées localement aux coordonnées différentielles et, enfin, le maillage prenant en compte les rotations est calculé.

Shi *et al.* [SYBF06] calculent un champ harmonique pour interpoler les rotations locales et utilisent un solveur hiérarchique pour accélérer les calculs.

En plus du besoin de résolution de gros systèmes linéaires en temps réel, les approches précédentes, linéaires, présentent des résultats de qualité moyenne pour des déformations importantes ; les rotations sont approchées et des artefacts peuvent apparaître pour des déformations importantes.

Les besoins en animation de personnages ont également fait naître l'utilisation de squelettes de manipulation pour remplacer les grilles de contrôle. Dans un premier temps le squelette est déformé, puis, chaque point de la surface initiale subit une transformation déduite des transformations du squelette. Shi *et al.* [SZT+07] présentent une approche basée sur un squelette représenté par des tétraèdres. Le squelette est déformé par une énergie non linéaire permettant de contrôler la forme déformée. Le problème est formulé sous forme variationnelle et un solveur hiérarchique complexe est utilisé.

Récemment, des approches non-linéaires, physiquement réalistes mais moins complexes à mettre en place que les approches basées sur les éléments finis, ont émergé. Sorkine et Alexa [SA07] notent que, lors d'une déformation, l'utilisateur souhaite que la forme manipulée soit localement préservée. Tout petit morceau de surface doit rester le plus rigide possible et les déformations doivent donc être des transformations rigides *i.e.* la composée d'une rotation et d'une translation. Pour ce faire, la surface à manipuler est recouverte de cellules définies au niveau des sommets et se chevauchant. Les auteurs utilisent une approche différentielle qui permet de gérer intrinsèquement les translations. La difficulté devient alors la détermination des rotations.

Après définition de contraintes sur certaines parties du modèle, la rotation de chaque cellule est calculée par analyse de la matrice de covariance des cellules libres et contraintes. La position spatiale de chaque cellule (*i.e.* chaque sommet) est obtenue en minimisant une énergie de rigidité au moyen d'une factorisation de Cholesky avant d'itérer le processus.

Sumner *et al.* [SSP07] proposent de construire un graphe de déformation, assimilable à un polygone de contrôle, sur les données à traiter et de façon que les sommets du graphe soient les mieux répartis possible sur la surface sous-jacente. Après introduction de contraintes sur certains sommets du graphe de déformation, une énergie globale est calculée comme somme de trois termes :

1. *une énergie de rotation*, définie pour chaque sommet du graphe, de façon à assurer des déformations les plus rigides possibles ;

2. *une énergie de régulation*, définie pour chaque sommet du graphe en prenant compte de son voisinage, qui simule un matériau le plus élastique possible afin d'avoir des déformations naturelles ;

3. *une énergie de contrainte*, définie pour chaque poignée, qui assure que les données proches d'un noeud contraint du graphe satisfont la contrainte imposée par l'utilisateur.

L'énergie globale du système, somme pondérée de ces trois énergies, est minimisée par la méthode de Gauss-Newton et les positions finales des données déformées sont calculées comme combinaison des transformations des sommets du graphe de déformation proches.
Botsch *et al.* [BPGK06, BPWG07] ont successivement proposé deux approches fondamentalement volumiques calculant des déformations rigides. Les deux approches diffèrent au niveau du domaine de définition. La première approche considère uniquement un petit volume centré sur la surface à manipuler là où la seconde modélise l'ensemble du volume définit par la surface manipulée. Les énergies considérées sont donc différentes. Dans la mesure où l'on souhaite principalement manipuler des nuages de particules volumiques, nous écartons la première approche pour ne considérer que la seconde. L'idée est de calculer une transformation rigide pour chaque voxel. Les auteurs définissent ainsi une énergie entre des voxels voisins et calculent les transformations à appliquer par une minimisation itérative comprenant deux étapes :

1. un problème global avec linéarisation des rotations ;
2. un problème local où chaque transformation linéarisée est projetée sur l'espace des transformations rigides.

Une fois la convergence atteinte, chaque voxel représentant le volume déformé a subi une transformation affine et on peut calculer, pour tout point de départ, une image par une transformation rigide. Une interpolation par fonction à base radiale est utilisée afin d'obtenir une fonction de déformation spatiale de continuité $\mathcal{C}^2$.

Discussion

L'outil recherché de déformation globale doit répondre à quatre attentes principales :

1. manipulation directe du modèle ;
2. déformation physiquement plausible, en temps réel ;
3. être agnostique quant à la représentation des données manipulées ;
4. robustesse pour ne pas limiter l'utilisation à certaines déformations.

On peut ainsi d'ores et déjà écarter certaines approches. Les approches de type *Free Form Deformation* (FFD) répondent au deuxième point mais ne répondent pas à notre premier besoin de manipulation directe. La solution proposée par Hsu *et al.* corrige artificiellement ce problème. Néanmoins le défaut fondamental de ces approches est la dé-corrélation entre la grille de déformation et la forme manipulée. Sans le travail manuel proposé par Coquillart, deux objets très différents seront traités par la même grille de contrôle. Les déformations produites pourront donc paraître peu naturelles.

Les approches basées sur le maillage sont trop liées à un type de représentation de surface. Les approches type éléments finis nécessitent un maillage, surfacique ou volumique, et ne sont, aujourd'hui, pas temps réel.

L'utilisation d'un squelette de déformation semble approprié pour la manipulation de formes représentant des personnages. Cependant, si on prend une forme quelconque, le squelette utilisé, par exemple l'axe médian du modèle, ne sera pas toujours très intuitif et pourra conduire à des résultats inattendus.

La gestion fine des rotations passe par une approche non linéaire. Nous avons présenté trois approches non-linéaires qui présentent le même objectif : déterminer des transformations localement rigides. Ces trois méthodes utilisent une représentation intermédiaire des données, calculent les transformations affines à appliquer à cette représentation et en déduisent les transformations à appliquer à la forme initiale :

1. [SA07] utilise un recouvrement de la surface par des petites cellules centrées sur les sommets initiaux (modélisation surfacique purement) ;
2. [SSP07] crée un graphe de déformation où les sommets du graphe sont obtenus en évaluant régulièrement la surface et les arêtes relient des noeuds proches (modélisation surfacique pouvant être étendue au volumique) ;
3. [BPWG07] est basé sur une voxelisation du modèle (modélisation volumique uniquement).

L'approche [BPWG07] est la plus générique et vraisemblablement la plus efficace pour notre problématique. Elle est naturellement volumique et, de par la structure d'arbre, permet un contrôle adaptatif pour affiner localement la représentation utilisée ou, au contraire, pour prendre une représentation plus grossière et alléger les calculs.

La difficulté est ensuite l'obtention d'une transformation rigide, le point délicat étant d'avoir une *vraie* rotation. Les trois méthodes proposées, là encore, diffèrent :

1. [SA07] propose une estimation de matrice de rotation basée sur la décomposition en valeurs singulières de la matrice de covariance des cellules puis en déduit les positions des sommets ;
2. [SSP07] utilise une énergie de rotation définie comme la distance d'une donnée à une matrice orthonormale dont les colonnes sont de norme unité et deux à deux orthogonales ;
3. [BPWG07] impose d'avoir une transformation rigide en relâchant le problème et en cherchant une solution au problème linéarisé puis projette la solution relâchée sur l'espace des transformations rigides en déterminant la matrice de rotation la plus proche de la matrice trouvée pour le problème linéarisé.

La solution utilisée par [SSP07] est clairement la plus permissive puisqu'à aucun moment on ne cherche à avoir une rotation mais on cherche simplement une matrice proche d'une matrice de rotation ; on peut voir cela comme la recherche d'une transformation *si possible rigide*. L'approche proposée dans [SA07] est plus forte puisqu'on commence par calculer une matrice de rotation mais le problème est ensuite relâché dans le calcul des positions des cellules qui prend en compte l'estimation calculée mais n'impose pas une transformation rigide et l'objectif est donc de trouver une transformation *la plus rigide possible*. [BPWG07] a une démarche inverse puisque le problème est d'abord relâché et linéarisé puis les solutions sont projetées sur l'espace des transformations rigides ; on obtient ainsi une transformation *rigide*.

Enfin, l'énergie globale du système diffère encore pour chaque approche :

1. [SA07] utilise une énergie élastique exprimée de façon différentielle

$$E_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} ||(\mathrm{x}'_i - \mathrm{x}'_j) - \mathbf{R}_i(\mathrm{x}_i - \mathrm{x}_j)||^2$$

   où $\mathrm{x}'_.$ désigne les coordonnées de la cellule $\mathrm{x}_.$ après déformation la plus rigide possible ;

2. [SSP07] définit une énergie élastique

$$E_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} ||\mathbf{R}_i(\mathrm{x}_j - \mathrm{x}_i) + \mathrm{x}_i + \mathrm{t}_i - (\mathrm{x}_j + \mathrm{t}_j)||^2$$

entre les sommets du graphe de déformation ;

3. de façon à ne pas dépendre d'un échantillonnage particulier mais de prendre en compte complètement le volume représenté par un voxel particulier, [BPWG07] propose d'utiliser l'énergie

$$E_i = \sum_{j \in \mathcal{N}(i)} \int_{C_i \cup C_j} ||\mathrm{t}_i + \mathrm{R}_i(\mathrm{x}) - \mathrm{t}_j - \mathrm{R}_j(\mathrm{x})||^2 d\mathrm{x}.$$

L'objectif est ainsi d'obtenir une transformation globale la plus élastique possible *i.e.* que deux transformations voisines soient les plus proches possibles.

Les approches [SA07, SSP07] sont relativement semblables et simulent, en quelque sorte, un matériau *discret* (*i.e.* reposant sur les cellules ou les sommets du graphe de déformation) élastique. Pour ne pas que la quantité d'information réellement contenue dans un voxel influe fortement sur le résultat de la déformation, [BPWG07] considère l'ensemble du volume de chaque voxel et simule une élasticité dans les transformations utilisées. Cette subtile nuance va permettre d'avoir des déformations les plus rigides possibles au niveau des voisinages de voxels et donc, comme évoqué par Sorkine et Alexa, d'obtenir une déformation plaisante sur l'ensemble de la forme.

Il serait possible de mélanger les trois approches et, par exemple, d'utiliser l'approche de détermination des rotations de [SA07] aux voxels de [BPWG07]. Cependant, pour tous les points évoqués, [BPWG07] est l'approche la plus générique du fait de l'utilisation d'une structure spatiale et la plus robuste car elle est basée sur des déformations parfaitement rigides et ne dépendant pas d'une quantité d'information discrète.

Nous avons donc retenu cette approche et allons présenter nos apports aux différentes étapes de la méthode.

6.2 Déformation globale par cellules rigides

L'approche repose sur une voxelisation de l'espace délimité par le modèle, rendant ainsi possible le stockage d'une surface ou d'un nuage de points. Seule la construction de la voxelisation pourra être différente selon le type de données en entrée. Les voxels, que nous supposons cubiques par commodité, peuvent ainsi être vus comme une méta-représentation de la forme manipulée.

Une fois la voxelisation déterminée, l'utilisateur va imposer des contraintes sur le modèle et ces contraintes sont alors appliquées aux voxels correspondants. Le système minimise une énergie de type élastique satisfaisant ces contraintes et détermine une transformation rigide pour chaque voxel non contraint. La géométrie du modèle peut alors être mise à jour en utilisant les transformations des voxels.

Nous noterons T une transformation rigide des $\mathbb{R}^3$, définie comme l'application d'une rotation R et d'une translation t *i.e.*

$$\mathrm{T}(\mathrm{x}) = R\mathrm{x} + \mathrm{t}.$$

Présentons à présent le détail de chaque étape.

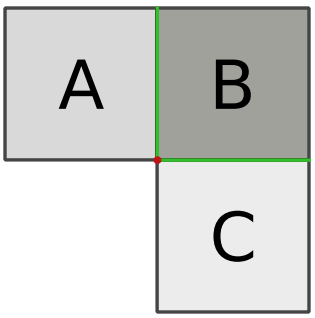


FIG. 6.1: Définition du voisinage de voxels. Le voxel B est voisin des voxels A et C mais les voxels A et C ne sont pas voisins.

6.2.1 Voxelisation

L'approche étudiée est basée sur les voisinages de voxels où deux voxels sont considérés comme voisins s'ils présentent une intersection surfacique non vide et non réduite à une arête ou un point[1] (fig. 6.1). On souhaite dans notre cas avoir une voxelisation indépendante du modèle sous-jacent.

En effectuant une voxelisation par descente *i.e.* un découpage récursif d'un volume racine englobant à concurrence du nombre maximal d'éléments stockés dans un voxel, la voxelisation dépendra fortement de l'échantillonnage de l'objet représenté. Nous avons donc opté pour une voxelisation par remontée en deux étapes.

On commence par évaluer le volume occupé par l'objet en approchant la fonction indicatrice sur une grille englobant le modèle (fig. 6.2b). On utilise les stratégies d'évaluation de la fonction caractéristique définies dans le chapitre 4.

Une fois la fonction indicatrice du volume évaluée, on divise l'octree à son niveau le plus fin et on ne conserve que les voxels dont l'intersection avec la fonction indicatrice est non nulle (fig. 6.2c).

L'utilisation d'une structure d'arbre pour représenter l'objet est intéressante car elle permet une adaptation locale, augmentant ou diminuant le nombre de degrés de liberté et permettant donc un ajustement à la volée. Par ailleurs, la déformation impactera d'abord les volumes proches de la surface du modèle avant d'atteindre le coeur de l'objet. Les voxels situées à l'intérieur du volume sont condensés pour diminuer le nombre de voxels représentant l'intérieur (fig. 6.2d).

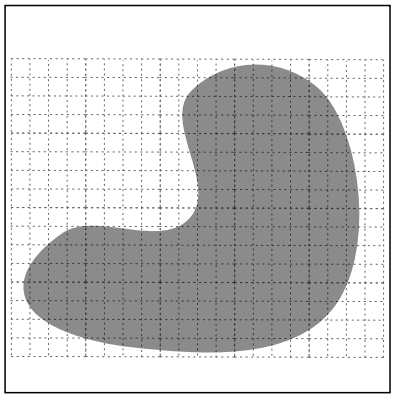
6.2.2 Contraintes

Lors d'une déformation, l'utilisateur impose des contraintes sur le modèle puis le système détermine les transformations à appliquer aux voxels libres pour satisfaire au mieux ces contraintes.

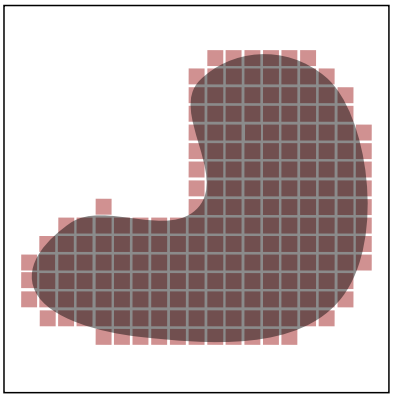
Initialement, Botsch *et al.* considèrent 3 types de contraintes sur les voxels :

- *voxel libre* : aucune contrainte n'est imposée au voxel ;
- *voxel fixe* : ne subit aucune transformation ni rotation ;

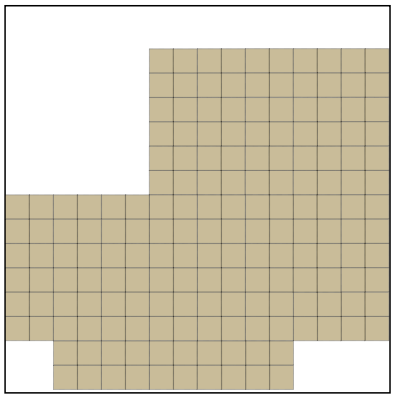
[1] De façon plus générale, deux voxels de $\mathbb{R}^n$ sont voisins si leur intersection est de dimension $n-1$ exactement.



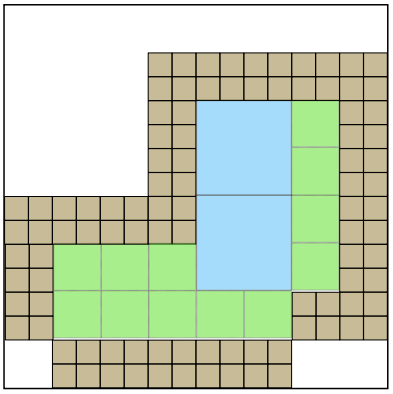
(a) Forme à traiter



(b) Évaluation de fonction caractéristique.



(c) Construction des feuilles de l'octree.



(d) Condensation des voxels intérieurs.

FIG. 6.2: Schéma de voxelisation.

- *voxel poignée* : la transformation subie par le voxel est imposée par l'utilisateur.

On peut constater que les rôles de voxel poignée et de voxel fixe sont en fait identiques puisque, dans chacun de ces deux cas, l'utilisateur spécifie intégralement la transformation associée à un voxel. La différence faite par Botsch *et al.* permet d'avoir un critère pour lancer le calcul d'une déformation puisque sans manipulation de poignée, le système n'est pas « perturbé », et il est donc inutile de rechercher une déformation. Nous avons ainsi réduit la sémantique au plus simple et considéré les deux catégories de contraintes suivantes :

- *voxel libre* : translation et rotation libres ;
- *voxel contraint* : translation et rotation fixées.

Notons qu'il serait aisé d'enrichir ces comportements de voxels en permettant par exemple de ne fixer qu'une partie de la transformation et de créer ainsi des voxels pivots (translation fixée, rotation libre) ou des voxels glissières (translation libre, rotation fixée) mais nous avons préféré simplifier au plus les possibilités de l'utilisateur.

6.2.3 Minimisation de l'énergie de déformation

Une fois les contraintes établies sur l'ensemble des voxels (un voxel étant libre par défaut), on détermine les transformations à appliquer à chaque voxel de façon à minimiser une énergie de déformation due aux déplacements des voxels contraints. Botsch *et al.* proposent d'utiliser une énergie de type élastique sur les transformations de voxels voisins et mesurent la distance des transformations appliquées à l'union des voxels :

$$E = \sum_{\{i,j\}} \frac{w_{ij}}{V_i + V_j} \int_{C_i \cup C_j} ||\mathrm{T}_i(\mathrm{x}) - \mathrm{T}_j(\mathrm{x})||^2 d\mathrm{x} \tag{6.1}$$

où V_i désigne le volume du voxel en configuration de référence C_i[2], T_i désigne sa transformation rigide associée, et w_{ij} est le poids attribué à la paire de voxels voisins $\{i, j\}$. Ce poids permet de privilégier certaines zones lors de la minimisation et permet ainsi de contrôler, dans une certaine mesure, la rigidité locale de la déformation. Par défaut, ce poids est égal au ratio de l'aire d'intersection entre les voxels voisins sur la distance séparant leur centre respectif de façon à prendre en compte le volume modélisé par chaque voxel.

Il est important de noter que l'énergie ne repose que sur la position des voxels et nullement sur les informations stockées par chaque voxel puisque l'intégrale est définie sur le volume du voxel et non sur l'échantillon d'information qu'il contient. L'énergie est trivialement nulle si l'ensemble des voxels subit la même transformation ($\forall i, \mathrm{T}_i = \mathrm{T}$).

Botsch *et al.* [BPGK06, BPWG07] proposent un schéma de minimisation de type Newton : le minimum du problème linéarisé est estimé puis la solution est projetée sur l'espace des transformations rigides.

6.2.3.1 Linéarisation

Supposons qu'une transformation rigide T_i définie par

$$\mathrm{T}_i(\mathrm{x}) = \mathrm{R}_i\mathrm{x} + \mathrm{t}_i,$$

s'applique à chaque voxel i du système. On perturbe le système en introduisant une nouvelle déformation $\delta\mathrm{T}_i$ qui donne alors les nouvelles transformations

$$\tilde{\mathrm{T}}_i(\mathrm{x}) = \delta\mathrm{T}_i \circ \mathrm{T}_i(\mathrm{x}) = \delta\mathrm{T}_i\Big(\mathrm{T}_i(\mathrm{x})\Big).$$

En faisant l'hypothèse que la transformation $\delta\mathrm{T}_i$ est « petite » , sa linéarisation $\mathcal{L}(\delta\mathrm{T}_i(\mathrm{x}))$ fournit une bonne approximation (voir Annexe B) *i.e.*

$$\mathcal{L}(\delta\mathrm{T}_i(\mathrm{x})) = \mathrm{x} + (\omega_i \wedge \mathrm{x}) + \mathrm{v}_i \approx \delta\mathrm{T}_i(\mathrm{x}), \tag{6.2}$$

où ω_i est équivalent à une vitesse angulaire et v_i est équivalent à une vitesse linéaire. La nouvelle transformation appliquée au voxel i peut alors s'écrire

$$\tilde{\mathrm{T}}_i(\mathrm{x}) = \delta\mathrm{T}_i \circ \mathrm{T}_i(\mathrm{x}) \approx \mathrm{T}_i(\mathrm{x}) + \omega_i \wedge \mathrm{T}_i(\mathrm{x}) + \mathrm{v}_i.$$

L'expression obtenue est linéaire sur les composantes du couple $(\omega_i, \mathrm{v}_i) \in \mathbb{R}^6$ ce qui permet ainsi de relâcher le problème au niveau de la rotation. Nous verrons par la

[2]La configuration de référence est déterminée par la voxelisation et est donc fixe dans le temps. Le système déformé s'obtient alors simplement par application des transformations rigides aux voxels associés *i.e.* $\mathrm{T}_i(C_i)$.

suite qu'il est aisé de retrouver une transformation rigide à partir de la transformation linéarisée.

Dans le cas où la transformation appliquée n'est pas petite, l'idée va être d'effectuer une décomposition en un ensemble de petits déplacements dont on commencera par chercher le linéarisé avant d'effectuer une projection sur l'espace des transformations rigides afin d'assurer la conservation des rotations. On peut ainsi écrire

$$\delta \mathrm{T}_i = \lim_{k\to\infty} \delta\mathrm{T}_i^{(k)} \circ \delta\mathrm{T}_i^{(k-1)} \circ \cdots \circ \delta\mathrm{T}_i^{(1)} \circ \mathrm{Id}.$$

On notera, de la même façon,

$$\mathrm{T}_i^{(k)} = \delta\mathrm{T}_i^{(k)} \circ \underbrace{\delta\mathrm{T}_i^{(k-1)} \circ \cdots \circ \mathrm{T}_i^{(0)}}_{\mathrm{T}_i^{(k-1)}}.$$

Pour prendre en compte la succession des déformations appliquées, on utilisera finalement la notation $\mathrm{T}_{i,n}^{(k)}$ qui correspond donc à la transformation du voxel i obtenue après k linéarisations à la n-ème déformation.

En utilisant l'approximation précédente, l'énergie à minimiser à la k-ème linéarisation de la n-ème déformation s'écrit

$$E_n^{(k)} = \sum_{\{i,j\}} \frac{w_{ij}}{V_i + V_j} \int_{C_i \cup C_j} ||\mathcal{L}(\delta\mathrm{T}_{i,n}^{(k)}) \circ \left(\mathrm{T}_{i,n}^{(k-1)}(\mathrm{x})\right) - \mathcal{L}(\delta\mathrm{T}_{j,n}^{(k)}) \circ \left(\mathrm{T}_{j,n}^{(k-1)}(\mathrm{x})\right)||^2 d\mathrm{x}.$$

En développant la norme, il vient

$$\begin{aligned}
& ||\mathcal{L}(\delta\mathrm{T}_{i,n}^{(k)}) \circ \left(\mathrm{T}_{i,n}^{(k-1)}(\mathrm{x})\right) - \mathcal{L}(\delta\mathrm{T}_{j,n}^{(k)}) \circ \left(\mathrm{T}_{j,n}^{(k-1)}(\mathrm{x})\right)||^2 \\
= \;& ||\mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) + \omega_{i,n}^{(k)} \wedge \mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) + \mathrm{v}_{i,n}^{(k)} - \mathrm{T}_{j,n}^{(k-1)}(\mathrm{x}) - \omega_{j,n}^{(k)} \wedge \mathrm{T}_{j,n}^{(k-1)}(\mathrm{x}) - \mathrm{v}_{j,n}^{(k)}||^2 \\
= \;& ||\mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) - \mathrm{T}_{j,n}^{(k-1)}||^2 \qquad (3) \\
+ \;& \left\langle \mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) - \mathrm{T}_{j,n}^{(k-1)}, \omega_i^{(k)} \wedge \mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) + \mathrm{v}_{i,n}^{(k)} - \omega_j^{(k)} \wedge \mathrm{T}_{j,n}^{(k-1)}(\mathrm{x}) - \mathrm{v}_{j,n}^{(k)} \right\rangle \qquad (2) \\
+ \;& ||\omega_{i,n}^{(k)} \wedge \mathrm{T}_{i,n}^{(k-1)}(\mathrm{x}) + \mathrm{v}_{i,n}^{(k)} - \omega_j^{(k)} \wedge \mathrm{T}_{j,n}^{(k-1)}(\mathrm{x}) - \mathrm{v}_{j,n}^{(k)}||^2 \qquad (1)
\end{aligned}$$

L'énergie est quadratique et les volumes des voxels étant bien définis, les intégrales volumiques sont calculables analytiquement. On se ramène à une écriture de l'énergie de la forme

$$E_n^{(k)} = \mathrm{p}^t B_n^{(k)} \mathrm{p} + \left(\mathrm{c}_n^{(k)}\right)^t \mathrm{p} + d_n^{(k)},$$

où :

- $B_n^{(k)}$ est une matrice carrée de taille $m = 6l$, avec l le nombre de voxels libres, encodant les relations quadratiques des couples (1),
- $\mathrm{c}_n^{(k)}$ est un vecteur de taille m encodant les relations linéaires des couples (2),
- $d_n^{(k)}$ est un scalaire encodant l'énergie du système après projection de la linéarisation précédente (3),
- et p est le vecteur de taille m des vitesses linéaires (v_i) et angulaires (ω_i) inconnues.

On peut noter que la matrice $B_n^{(k)}$ est symétrique puisque la relation de voisinage l'est. Par ailleurs, comme elle encode l'énergie, nécessairement positive ou nulle, de couples libres, elle est positive. Néanmoins, si tous les voxels subissent la même transformation, l'énergie du système est nulle et donc la matrice $B_n^{(k)}$ n'est pas bien définie. Enfin, du fait de la localité du voisinage, la matrice est creuse. Le nombre de coefficients dépend du nombre de couples de voxels voisins qui est petit devant le nombre total de voxels.

Par symétrie de la matrice $B_n^{(k)}$, le minimum de l'énergie peut alors s'obtenir en résolvant le système

$$B_n^{(k)}\mathrm{p} = -\frac{1}{2}\mathrm{c}_n^{(k)}.$$

Notons que tant que les rôles des voxels sont inchangés, la structure de la matrice reste la même et seuls les coefficients changent du fait de la modification des domaines d'intégration. Botsch *et al.* proposent donc d'utiliser des matrices creuses avec factorisation symbolique et de résoudre le système par la méthode de Cholesky [BPGK06].

6.2.3.2 Projection rigide

La minimisation de l'énergie précédente fournit, pour chaque voxel, le linéarisé $\mathcal{L}(\delta\mathrm{T}_i^{(k)})$ de chaque petit déplacement à appliquer. Afin de supporter des grandes déformations de façon robuste, ces transformations affines sont projetées sur l'espace des transformations rigides. Cette opération correspond à la transformation permettant de passer d'un repère orthonormé à un nouveau repère orthonormé. La difficulté est d'obtenir une matrice de rotation exacte. On renvoie à l'annexe B pour les détails de la méthode provenant des travaux de Horn [Hor87].

L'obtention de la matrice de rotation nécessite la recherche de la plus grande valeur propre d'une matrice symétrique de dimension 4×4. Nous avons retenu une approche de calcul du polynôme caractéristique par la méthode de Faddeev-Leverrier [SK00] puis recherche des racines par la méthode de Ferrari. En pratique le temps de calcul des projections des transformations linéarisées est négligeable devant le temps de minimisation du problème linéarisé. Notons que chaque voxel est indépendant des autres et que cette étape peut directement être parallélisée.

6.2.3.3 Analyse du mouvement

Ce schéma à deux étapes de relaxation-correction (*i.e.* minimisation du problème linéarisé-projection de la solution sur l'espace des transformations rigides) est répété jusqu'à convergence de l'énergie du système rigide.

Si le mouvement de déformation imposé par l'utilisateur est continu, on peut supposer que deux déformations successives sont proches ce qui justifie l'utilisation de la linéarisation. Cette hypothèse de mouvement continu peut être utilisée plus fortement pour la minimisation de l'énergie de deux déformations successives.
Naïvement, le processus de minimisation devrait être effectué en partant du modèle initial. Ceci impliquerait un nombre croissant de linéarisations à mesure que la déformation devient plus importante. L'hypothèse du mouvement continu permet d'utiliser le résultat de la $(n-1)$-ème déformation comme point de départ pour la minimisation de la n-ème déformation. Il s'agit concrètement de conserver les transformations de la déformation $(n-1)$-ème pour les voxels libres et d'appliquer

les transformations imposées par l'utilisateur pour les voxels contraints. Dans le cas où le mouvement ne serait pas continu, l'utilisation de cette hypothèse entraînerait un nombre supérieur de linéarisations pour atteindre le minimum.

L'autre hypothèse effectuée est celle de *petits déplacements*. En effet, la linéarisation des rotations nécessite de *petits* angles pour être valide. Le vecteur des transformations linéarisées devrait donc être généralement proche du vecteur nul. Cette hypothèse n'est pas exploitable avec une méthode de résolution directe puisque ces méthodes reposent uniquement sur la donnée d'une matrice et d'un second membre et que la résolution ne peut être guidée. Une méthode itérative pourrait cependant exploiter cette hypothèse efficacement.

Synthétisons notre analyse :

- *hypothèse du mouvement continu* : à la construction du premier système $B_n^{(0)}$ de la n-ème déformation, on a

$$T_{i,n}^{(0)} \quad = \quad T_{i,n-1}^{(\infty)}. \tag{6.3}$$

  L'hypothèse est valable pour toute méthode de résolution.

- *hypothèse de petits déplacements* : à chaque système, pour valider la linéarisation effectuée on peut supposer le vecteur solution comme proche du vecteur nul et donc considérer comme vecteur initial

$$\mathrm{p}_0 = 0. \tag{6.4}$$

  L'hypothèse n'est utilisable qu'avec une méthode de résolution itérative.

6.2.3.4 Résultats

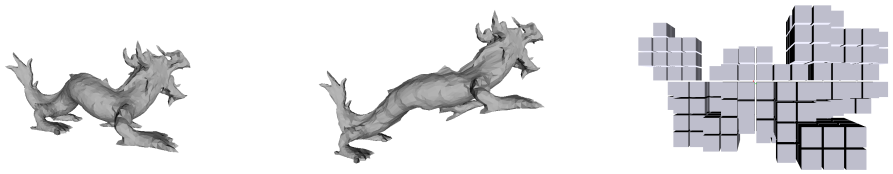
Nous avons précédemment vu que la matrice $B_n^{(k)}$ n'est pas bien définie. Cependant, la forme bilinéaire s'annule uniquement dans le cas d'une transformation rigide uniforme sur tous les voxels soit

$$\forall i, \mathrm{T}_i = \mathrm{T} \Rightarrow \mathrm{p}^t B_n^{(k)} \mathrm{p} = 0.$$

L'utilisateur impose généralement une partie fixe et contraint une autre partie ; la solution ne sera alors pas une transformation homogène sur l'ensemble du modèle. Le cas où une seule contrainte est imposée (elle peut être imposée sur une ou plusieurs parties du modèle) peut être problématique. Ce cas est détectable et peut être traité directement puisque la transformation des voxels libres correspond alors à la contrainte imposée. Le problème pourrait également apparaître au cours de la résolution mais il est également détectable. Nous avons donc utilisé une résolution du système par gradient conjugué.

Afin de mesurer l'efficacité de cette approche et des hypothèses du mouvement continu et des petits déplacements, nous avons comparé de l'utilisation de la méthode du gradient conjugué avec la méthode de Cholesky. Chacune des méthodes a été implémentée sur des matrices creuses reposant sur un stockage aléatoire [LT00].

Pour étudier nos remarques précédentes liées au mouvement, 100 petites déformations, sont effectuées (fig. 6.3). L'énergie est minimisée par la méthode de Cholesky et celle du gradient, en faisant ou non l'hypothèse de déplacement continu. L'implémentation du gradient conjugué utilise, dans les deux cas, l'hypothèse



(a) Modèle initial.

(b) Modèle après 100 déformations.

(c) Voxelisation du modèle initial.

FIG. 6.3: Scénario de test pour la minimisation d'énergie.

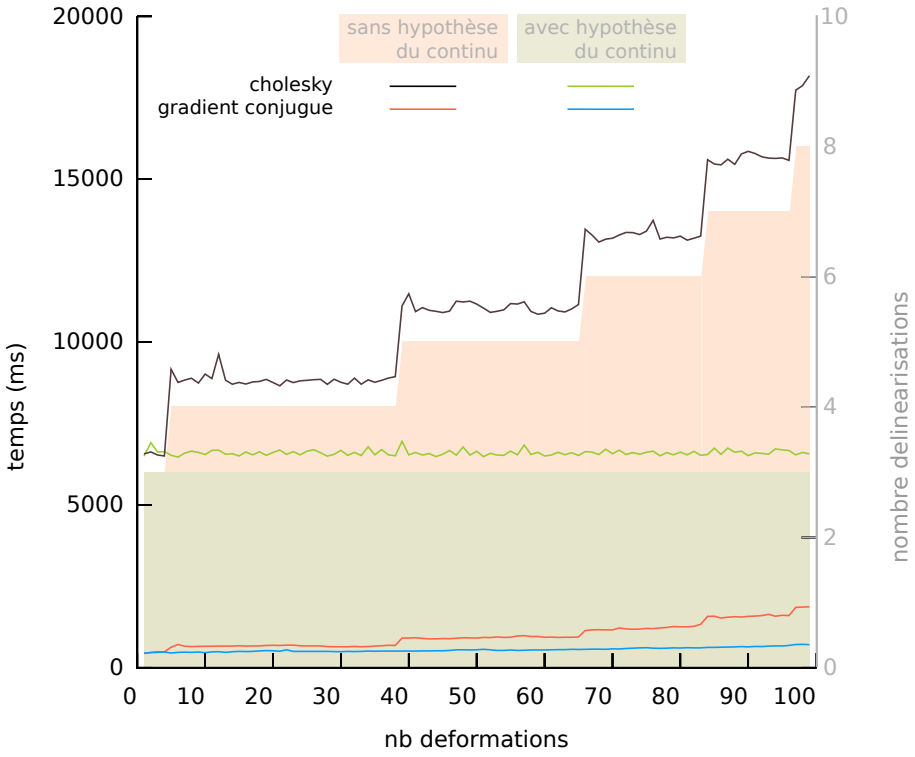


FIG. 6.4: Comparaison des méthodes de Cholesky et du gradient conjugué pour la minimisation d'énergie.

des petits déplacements donc prend comme point initial le vecteur nul. On utilise comme critère d'arrêt une variation relative de l'énergie de $\frac{1}{1000}$.

Les écarts relatifs des énergies obtenues après les 100 itérations sont d'au plus 1% et tous les résultats finaux sont visuellement identiques. Ceci permet de valider les résultats obtenus par le gradient conjugué. Par ailleurs, la norme infinie des vecteurs solution de chaque linéarisation est au plus de l'ordre de 10^{-3}. Les déplacements calculés sont donc bien petits.

Les résultats (fig. 6.4) mettent en avant la corrélation entre le temps de résolu-

tion et le nombre de linéarisations effectuées. Pour des déformations nécessitant un nombre équivalent de linéarisations, le temps requit par la minimisation est quasi-constant.

Le nombre de linéarisations effectuées pour une déformation devient constant avec l'hypothèse du mouvement continu. Par conséquent, cette hypothèse permet de garantir, pour toute déformation effectuée de façon continue, un temps de résolution quasiment constant, même pour des déformations importantes effectuées de façon continue.

On peut finalement noter que le temps d'exécution du gradient conjugué permet de gagner un ordre de grandeur sur le temps de résolution. Il apparaît donc avantageux d'utiliser les deux hypothèses proposées et ainsi d'utiliser une méthode itérative de résolution à l'instar du gradient conjugué pour obtenir des résultats interactifs.

Comme nous l'avons vu, pour sembler naturelle, une déformation doit être localement la plus rigide possible. Ainsi, plus des voxels considérés sont distants, et plus les variations de leurs transformations sont tolérées. Cette remarque peut servir à la mise en place d'une méthode multi-résolution.

En utilisant la structure de découpage hiérarchique, il serait possible de minimiser l'énergie sur des problèmes de tailles croissantes, en descendant progressivement la structure de découpage et en minimisant l'énergie pour chaque niveau. Un voxel supérieur devient contraint si, dans son arborescence, il contient un voxel contraint. Si l'arborescence contient plusieurs contraintes différentes, on peut utiliser la contrainte moyenne.

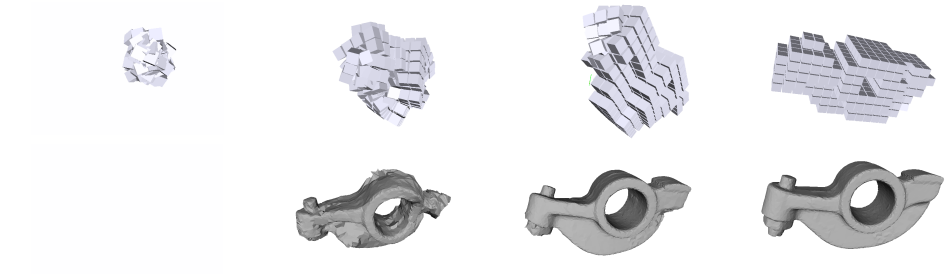
En descendant cette structure, chaque sous-voxel subirait une transformation rigide identique à ses voxels voisins, produisant une rigidité locale. À chaque nouvelle résolution, la déformation locale serait affinée et relâcherait la rigidité entre voxels voisins. Ce schéma de minimisation multi-résolution semble naturel et performant. Néanmoins, l'hypothèse du mouvement continu assure un nombre réduit de linéarisations et celle des petits déplacements permet une convergence rapide de la méthode du gradient conjugué. L'approche multi-résolution proposée ne semble donc pas nécessaire.

Enfin, nous avons réalisé le cas limite de contraction de tous les voxels d'un modèle en un même point avec des rotations aléatoires sur chaque voxel, un seul restant fixe. Cette transformation correspond à une déformation non continue. Les résultats obtenus (fig. 6.5) montrent la robustesse de la minimisation avec utilisation de la méthode du gradient conjugué.

6.2.4 Reconstruction

Une fois l'énergie minimisée, on dispose de la transformation associée à chaque voxel. Il faut alors déduire la position des données à partir de celles des voxels. Seule la géométrie est modifiée, la topologie est conservée.

La voxelisation englobe l'ensemble du volume défini par le modèle manipulé. On peut alors associer un repère local à chaque voxel. La reconstruction peut donc simplement consister à appliquer la déformation de chaque voxel à l'ensemble des données qu'il contient. On opérerait donc bien une transformation rigide des données. Néanmoins, cette stratégie peut évidemment produire des discontinuités puisque deux échantillons de données situés de part et d'autre de la frontière d'un couple de voisins subiront des transformations d'autant plus différentes que le couple aura



(a) Pertubation des 400 voxels. (b) 5 linéarisations. (c) 10 linéarisations. (d) 25 linéarisations.

FIG. 6.5: Robustesse du schéma de minimisation.

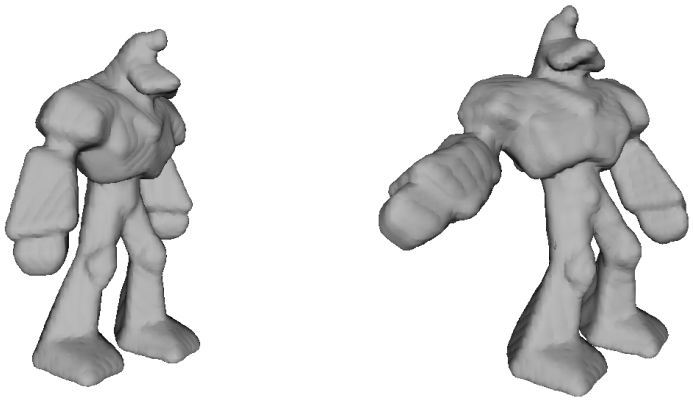


FIG. 6.6: Déformation globale d'un système de particules.

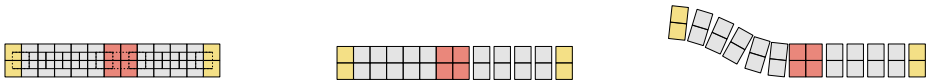
une énergie importante. D'autre part, l'interpénétration des voxels n'est pas explicitement interdite ; aussi, il se peut qu'une reconstruction naïve entraîne des retournements de triangles dans le cas de la déformation d'un maillage triangulaire.

Botsch *et al.* proposent alors deux solutions. La première, utilisée en temps réel, consiste à appliquer une combinaison des transformations rigides des voxels voisins. La deuxième est la reconstruction d'une fonction de déformation continue sur l'ensemble de l'espace manipulé, calculée à partir de fonctions à base radiale.

Notre but premier est la déformation de nuages de particules qui ne modélisent pas les hautes fréquences ; les artefacts visibles sur des surfaces très détaillées ne sont pas applicables dans notre contexte. Nous nous sommes donc contentés d'utiliser une reconstruction par combinaison de transformations rigides en utilisant comme poids la distance aux centres des voxels voisins (fig. 6.6).

6.2.5 Déformation semi-globale

La notion de voisinage de voxels induit une topologie. On peut représenter cette topologie par un graphe d'interaction dont les sommets sont les voxels et une arête existe entre deux sommets s'ils sont voisins et ne sont pas tous les deux contraints.



(a) Graphe d'interaction (en pointillé les interactions supprimées par contraintes).

(b) Déformation d'une moitié de modèle non propagée à l'ensemble.

(c) Déformation de l'autre moitié du modèle sans impact sur la première moitié.

FIG. 6.7: Déformation semi-globale.

Il s'agit donc d'un graphe analogue au graphe dual de la voxelisation dans lequel deux voxels contraints ne sont pas reliés entre eux puisque, par définition, ils sont contraints et ne peuvent interagir entre eux.

Chaque contrainte induit donc une extrémité dans le graphe d'interaction (fig. 6.7 gauche). On va pouvoir exploiter cette propriété pour limiter la taille des problèmes à traiter. En effet, si l'on observe l'effet des contraintes dans le cas d'une barre contrainte en ses extrémités et en son milieu, on constate que la modification d'une extrémité n'a d'influence que sur une moitié de barre du fait de la présence d'une zone contrainte au milieu (fig. 6.7b et 6.7c).

Les contraintes peuvent donc créer différentes parties connexes sur le graphe d'interaction et toute contrainte ne peut avoir d'impact que sur une partie connexe. Ainsi, en construisant le graphe d'interaction et, à chaque modification des parties contraintes, en calculant les différentes partie connexes du graphe, on est capable de réduire la taille des systèmes à résoudre.

Des stratégies automatiques de diminution de la taille des problèmes à traiter sont envisageables. Lorsque l'utilisateur contraint une partie, la contrainte peut être diffusée sur les voxels voisins afin d'augmenter le nombre de composantes connexes du graphe d'interaction.

En attribuant un poids, par exemple simplement 1 ou alors une distance liée à la différence de niveau des voxels dans l'arbre, sur chaque arête du graphe, et en ayant un paramètre de distance λ de diffusion maximale, on peut marquer tous les voxels situés à au plus λ du voxel contraint. Si, en considérant tous ces voxels contraints, le nombre de composantes connexes du graphe d'interaction augmente, alors l'ensemble des voxels est contraint. Comme les graphes d'interaction sont constitués de peu de voxels, cette recherche peut s'effectuer en temps réel.

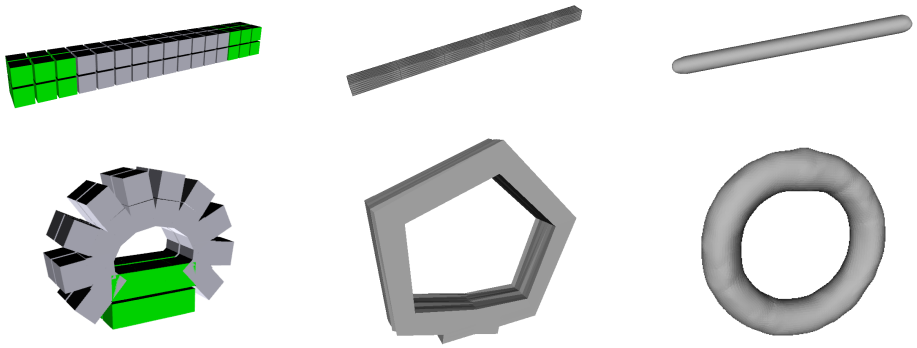
Notre stratégie présente deux avantages : plus le nombre de voxels contraints est grand, plus les nombre d'inconnues diminue. Et, en diminuant la dimension des problèmes à résoudre, les temps de calculs sont grandement améliorés.

6.2.6 Ré-échantillonnage local

L'approche de déformation globale présentée est performante. Elle répond par ailleurs pleinement aux attentes fixées précédemment :

- l'utilisateur peut manipuler directement le modèle qu'il souhaite et le déformer. Les contraintes imposées au modèle sont reportées sur la représentation voxel ;

- la déformation est calculée en temps réel ;



(a) Déformation de la voxelisation.

(b) Déformation sur une surface peu échantillonnée.

(c) Déformation d'un système de particules.

FIG. 6.8: Problème d'échantillonnage pour la déformation globale.

- la représentation du modèle par une voxelisation permet de gérer tout type de représentation géométrique à trois dimensions. L'utilisation de la méthode sur notre modèle de pâte à modeler le démontre (fig. 6.8c) ;

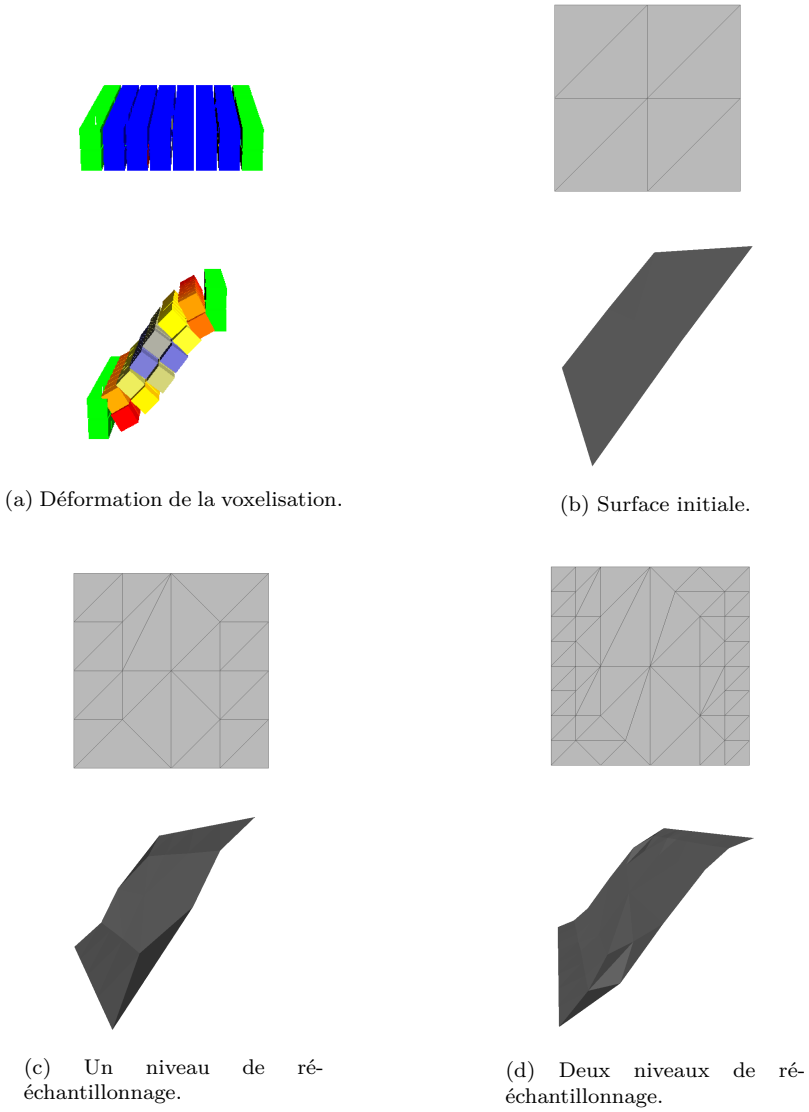
- le schéma de minimisation d'énergie par relaxation-projection est très performant. La figure 6.5 illustrant le cas limite consistant à compacter toute la géométrie aléatoirement en un point puis à minimiser l'énergie correspondante en atteste.

Dans notre contexte de création de forme, la puissance de la méta-représentation volumique a néanmoins le défaut d'être fortement décorrélée de l'échantillonnage du modèle utilisé. Ceci peut entraîner une perte d'information (fig. 6.8b). L'utilisation de système de particules, échantillonnant uniformément le volume de l'objet, peut résoudre le problème (fig. 6.8c), et permet également de gérer d'éventuels changements de topologie. Cependant, cette solution nécessite un changement peu souhaitable de modélisation de l'objet car la conversion en système de particules peut, elle aussi, entraîner des pertes d'information.

L'ensemble de la surface pourrait être ré-échantillonné finement ; cette solution a cependant le désavantage d'alourdir inutilement le modèle dans la plupart des endroits et, par conséquent, d'engendrer des temps de reconstruction de la surface plus importants. Il est donc intéressant de chercher à résoudre le problème d'échantillonnage des modélisations surfaciques localement.

Le problème intervient globalement si le nombre de voxels est supérieur au nombre d'échantillons. Dans ce cas, certains voxels apportent une information qui ne sera pas (ou peu, puisque chaque échantillon est déplacé selon un voisinage de voxels) utilisée. Plus particulièrement, il faut que les zones à forte énergie soient bien représentées car elles représentent des différences de transformations locales plus importantes.

En projetant les énergies des voxels sur les éléments de surfaces qu'ils contiennent, on crée un lien local entre la géométrie manipulée et sa méta-représentation. Associer à une primitive l'énergie des voxels contenant ses sommets uniquement ne suffit pas puisqu'elle ne prend pas nécessairement en compte l'ensemble des voxels situés au bord qui portent tous de l'information.



(a) Déformation de la voxelisation.

(b) Surface initiale.

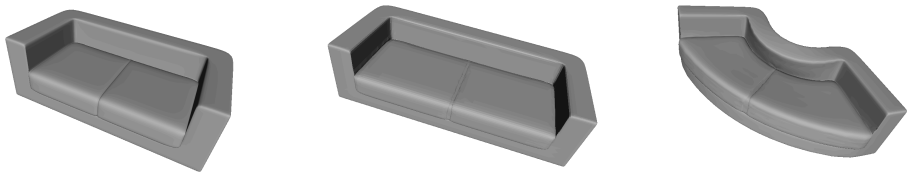
(c) Un niveau de ré-échantillonnage.

(d) Deux niveaux de ré-échantillonnage.

FIG. 6.9: Ré-échantillonnage local d'une surface par projection d'énergie.

Nous proposons ainsi d'associer aux primitives de surface l'énergie de tous les voxels qu'ils rencontrent *i.e.* un triangle aura pour énergie la somme des énergies des couples de voxels qu'il intersecte. Si l'énergie d'une primitive est supérieure à un seuil, alors la primitive est raffinée récursivement, tant que chaque nouvelle primitive introduite a une énergie supérieure au seuil fixé. Ce seuil a intérêt à être supérieur à la moyenne de des énergies de chaque voxel pour ne pas sur-échantillonner inutilement des éléments situés à l'intérieur d'un voxel.

À noter que, la déformation étant appliquée uniquement à la géométrie sans tenir compte de la topologie du modèle, il est nécessaire d'effectuer un raffinement des triangles qui reste *manifold*, sous peine de faire apparaître des trous dans la surface.



(a) Modèle original. (b) Modèle étiré. (c) Modèle courbé.

FIG. 6.10: Recherche de forme par déformation globale.

Dans le cas d'une translation pure, par exemple pour allonger une partie du modèle, il est possible que notre stratégie induise un sur-échantillonnage inutile. Il serait possible de ne prendre en compte que l'énergie liée aux rotations pour palier au problème mais ceci nécessiterait de recalculer une énergie sans que le gain global soit très important (fig. 6.9).

6.3 Conclusion

L'approche proposée par Botsch *et al.* [BPWG07] est très intéressante de par sa méta-représentation qui permet de gérer tout type de surface. Nous avons vu qu'elle répondait aux différents critères que nous avions imposés pour la déformation globale.

Nous avons apporté un certain nombre d'améliorations :

- l'utilisation d'une méthode de résolution itérative plus performante que celle initialement proposée,
- l'amélioration les résultats avec une recherche de composantes connexes dans un graphe d'interaction,
- l'application des déformation à des systèmes de particules,
- un mécanisme de ré-échantillonnage pour ne pas perdre d'information utile.

Les origines physiques de l'énergie utilisée produisent des formes globales naturelles et des résultats plaisants, notamment pour la recherche de forme sur des objets de la vie courante (fig. 6.10a).

Au delà de la seule déformation, l'approche permet également d'affiner une forme esquissée (fig. 6.10b), voire de créer de nouvelles formes (fig. 6.10c). L'intégration de cette approche permet d'enrichir le spectre de formes réalisables avec notre modèle particulaire. L'utilisation d'une méthode de ré-échantillonnage des modèles ou des systèmes de particules, permet de ne pas imposer le niveau de précision nécessaire en entrée.

Nous envisageons de coupler l'approche avec des outils de manipulation de surface de subdivision pour lesquelles on manipulerait le maillage de contrôle.

Nous avons privilégié d'autres stratégies d'optimisation, néanmoins il serait intéressant d'implémenter la méthode de minimisation multi-résolution évoquée dans la sous-section 6.2.3, et de tester l'utilisation de méthode de préconditionnement afin de pouvoir augmenter la précision des déformations calculables en temps réel.

L'approche s'applique également très bien à la déformation de personnages et il serait intéressant de chercher à lier les contraintes des voxels à des squelettes d'animation pour produire des animations de bonne qualité. L'approche permettrait alors de reconstruire en temps réel la « peau » du personnage (fig. 6.6).

Enfin, à l'instar de Hecker *et al.* [HRE+08], en couplant par exemple l'approche avec la recherche des parties significatives d'un modèle (sous-section 5.1.6), nous pourrions utiliser l'outil pour générer des animations automatiques sur des modèles quelconques.

Chapitre 7

Peinture et ajouts de détail

L'obtention d'une forme à proprement parlé n'est que la première étape du processus complet de création d'un modèle. Une fois la géométrie de la forme créée, l'artiste procède à la peinture et l'ajout de petits détails sur celle-ci pour lui donner plus de réalisme. Il est donc important de fournir des outils permettant d'ajouter de la couleur, voire des images, et des petits détails pour compléter les possibilités de création offertes à l'utilisateur.

Nous présentons ici quelques réflexions et un début d'implémentation des idées présentées.

7.1 Etat de l'art

Approches pour des surfaces paramétriques

L'ajout de détail à une surface peut se faire par l'ajout de couleurs ou d'images. Les premières applications d'images sur des surfaces remontent à Blinn et Newell [BN76] qui, pour améliorer le rendu des surfaces qui ne prenaient alors en compte que le calcul d'ombrage, combinent une méthode de filtrage et reprennent l'équation d'illumination de Phong pour effectuer un rendu de surfaces brillantes dans un environnement virtuel. Le rendu exact du placage d'une image sur une surface nécessite une évaluation correcte des normales et requiert donc de travailler sur des surfaces paramétriques et non simplement sur des polygones. Les surfaces ainsi rendues peuvent paraître artificiellement lisses. Pour augmenter encore le réalisme, Blinn [Bli78] introduit alors le *bump mapping* afin de donner une impression de relief. Les équations de rendu sont fortement basées sur la normale de la surface qui permet, en un point de la surface, de calculer une orientation par rapport à une source lumineuse. La modélisation de petits détails au niveau géométrique étant trop coûteuse, l'idée de Blinn est de déplacer chaque point artificiellement à une petite hauteur selon la normale à la surface et de calculer quelle serait alors la normale. En attribuant de cette façon une normale artificielle à chaque point de la surface on peut modéliser des petites « rides » sur la surface.

Pour gérer les problèmes d'évaluation des pixels écran après placage de la texture sur la surface, Williams [Wil83] introduit le *mip-mapping* « *multum in parvo* ». L'idée est d'utiliser une texture pyramidale où le niveau k de la pyramide contient la texture à appliquer en résolution $2^k \times 2^k$; en fonction du niveau de zoom sur l'objet, le système interpole différents niveaux de la texture pyramidale. Williams propose également d'étendre le concept à la géométrie et introduit déjà la notion de niveau de détails pour une surface qui attribuerait à chaque point d'un plan une

hauteur, soit une *height map*.

Cook [Coo84] étend le concept de *bump mapping* en perturbant directement la position des points de la surface et plus seulement leur normale via du *displacement mapping*.

Jusqu'alors, seules les approches de peinture en 2D sur ordinateur [Smi78, Smi79] permettaient une utilisation interactive, obligeant le placage de texture sur des surfaces tridimensionnelles à utiliser des images déjà existantes et à être non interactif. Dès lors que la puissance de calcul fut suffisante, un système interactif de décoration de surfaces quadrangulaires fut proposé par Hanrahan et Haeberli [HH90]. Le système réagit directement aux entrées utilisateur et crée en temps réel la texture appliquée sur la surface permettant ainsi de réellement peindre directement sur une surface tridimensionnelle sans passer explicitement par une image plane. Hanrahan et Haeberli proposent de définir une interface utilisateur autorisant différents types d'ajouts de détail : peinture de matériau (définissant les propriétés de rendu de la surface), peinture de couleur, peinture de petits détails. Si la souris ou un stylet permettent de simuler efficacement le travail d'un artiste en deux dimensions, ce n'est plus le cas lorsqu'on passe en 3D. Les auteurs proposent donc plusieurs façons de prendre en compte les entrées utilisateurs :

peinture dans l'espace paramétrique : la position du pinceau sur l'écran est associée à un point de la surface 3D qui est directement traduit en un point de la texture via ses paramètres ;

peinture par projection écran : l'utilisateur peint directement sur l'écran puis la texture produite est ensuite plaquée sur la surface ;

peinture dans l'espace tangent : l'utilisateur peint dans le plan tangent à la surface puis la texture est projetée sur la surface.

Les auteurs définissent également la notion de trait soit comme un ensemble discret de points (« *rubberstamp* »), soit, par interpolation linéaire, comme des lignes droites (« *rubberband* »).

Le système pour maillages quadrangulaires proposé par Williams [Wil90] est relativement proche ; chaque face du maillage est associée à une région d'une image 2D et séparée des autres faces du maillage par un espace nommé *gutter space*, de façon à éviter les problèmes d'échantillonnage de la texture. Étendant le concept de *height map*, Williams propose également d'utiliser la luminance des pixels d'une texture pour déterminer la hauteur des sommets de la surface et établit ainsi un système de sculpture.

Elkoura [Elk] présente quelques résultats de ces différents types de peinture et soulève quelques difficultés techniques comme les problèmes de performances d'association des coordonnées écran aux coordonnées 3D de l'objet et introduit également une notion de trait utilisant l'algorithme de Bresenham dans l'espace écran.

Plutôt que d'attribuer un morceau d'image à chacun des quadrangles d'une surface, Burley et Lacewell [BL08] proposent d'attribuer une texture par face pour permettre un rendu de très haute qualité. La problématique devient alors la gestion de l'application de pinceau à la frontière des quadrangles. L'avantage est que chaque face devient indépendante des autres et la forme du modèle peut donc être retravaillée sans pour autant devoir recréer toutes ses textures.

Approches pour des surfaces polygonales

Les techniques précédentes nécessitent d'avoir une représentation paramétrique de la surface à peindre. Les méthodes de paramétrage de surface polygonales sont apparues plus tardivement et les techniques précédentes ont donc longtemps été utilisées uniquement avec des surfaces paramétriques. Pour lever cette limitation, Agrawala *et al.* [ABL95] attribuent interactivement l'information de couleur directement aux sommets d'un maillage 3D. Pour éviter les artefacts visuels, une tessellation dynamique du maillage est réalisée en fonction du point de vue ; les triangles doivent représenter une surface inférieure à celle d'un demi pixel du rendu final. Pour accélérer la tessellation dynamique, Agrawala *et al.* proposent d'utiliser une table de hachage pour sélectionner les triangles visibles. Les pinceaux sont volumiques (boule, cylindre ou cône) et peuvent également agir directement sur la géométrie. Une notion de trait est également définie par simple interpolation linéaire de deux positions 3D consécutives du pinceau.

Les maillages polygonaux ayant gagnés en popularité grâce aux progrès matériels, la littérature s'est ensuite penchée sur la problématique du paramétrage de ces surfaces de façon à pouvoir utiliser les concepts développés pour les surfaces paramétriques. Les approches récentes se sont donc focalisées à trouver de bons paramétrages, conservant au mieux les propriétés de la surface tridimensionnelle. La conservation des longueurs étant impossible dans le cas général, Lévy *et al.* [LPRM02] utilisent des paramétrages conformes *i.e.* conservant au mieux les angles.

Igarashi et Cosgrove [IC01] utilisent des paramétrages écran ; tant que l'utilisateur ne change pas son point de vue, il ajoute du détail dans le plan écran. Une fois qu'il change de vue, le système détermine, dans la position fixe précédente, le paramétrage des faces qui se trouvaient sous les coups de crayon. Celle-ci correspond à leur projection écran et une nouvelle texture est créée. Si une face était déjà texturée, on projette la texture qui lui était associée sur le nouveau paramétrage et la nouvelle texture créée lui est attribuée. Cette approche permet de gérer en multi-résolution, des modèles non initialement paramétrés. Cette technique permet d'utiliser l'espace texture au mieux, en utilisant des pixels seulement là où c'est nécessaire.

Cependant, l'attribution d'une nouvelle texture à chaque point de vue peut s'avérer coûteuse en ressources. Carr et Hart [CH04] pré-calculent une segmententation hiérarchique de la surface de sorte que chaque segment puisse tenir au mieux dans une région carrée de la texture. La structure hiérarchique permet notamment de gérer du *mip-mapping*. Là encore un rendu par projection est utilisé et un reparamétrage est effectué en fonction d'une carte d'importance basée sur le gradient de couleur dans le texture.

Enfin Yuksel *et al.* [YKH10] utilisent un système basé sur la colorisation des éléments de base, sommets, arêtes, faces, qui peuvent être non uniformément réechantillonnés de façon à permettre un ajout localisé de détail. En jouant sur les différents échantillonnage, l'approche permet de gérer le *mip-mapping* et surtout ne nécessite pas de paramétrage de la surface manipulée et lève ainsi implicitement les problèmes de continuité rencontrés pour les surfaces non homéomorphes à un disque.

Approches volumiques et implicites

Plus récemment encore a émergé l'utilisation de textures 3D [DGPR02, BD02] qui permettent d'ajouter du détails directement dans l'espace. Les pigments ajoutés

sur la surface sont stockés directement en 3D sous la forme d'un octree. Le rendu final se fait vient un rendu dédié.

Schmidt *et al.* [SGW06] utilisent des applications exponentielles discrètes comme paramétrages locaux, permettant de travailler de façon souple et avec des outils identiques à ceux de l'édition d'image 2D traditionnelle sur tout type de surface pouvant être échantillonnée. L'application exponentielle envoie tout point de la surface dans l'espace tangent autour du point pour lequel elle est calculée. L'algorithme proposé repose sur une version modifiée de l'algorithme de Dijkstra, utilisé pour évaluer des géodésiques sur la surface.

Discussion

Cette brève revue de l'état de l'art nous montre que la littérature présente plusieurs types de solutions pour ajouter du détail à une surface :

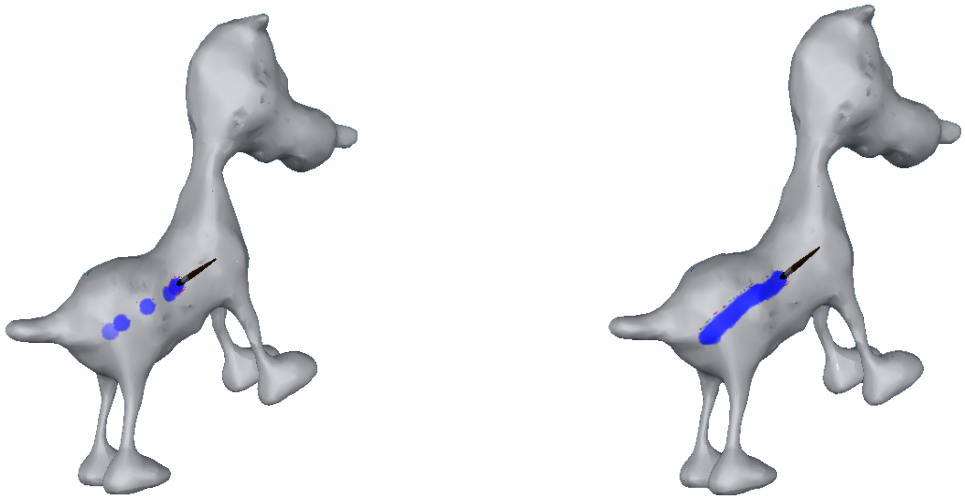
- l'utilisation de couleur associée à chaque point de la surface ;
- la perturbation, soit visuelle soit directe, des points de la surface.

Ces deux approches, qui peuvent sembler très différentes, ont été traitées avec succès par l'utilisation du placage de texture dont la nature dépend évidemment du résultat souhaité.

Nous souhaitons de nouveau un outil générique, permettant d'ajouter du détail à la fois sur des surfaces modélisées par nos surfaces implicites mais également sur des surfaces déjà existantes. Nous ne retiendrons pas les approches basées sur des fonctions implicites. De plus, si la création de la forme nous semble être réellement volumique, sa décoration s'appuie sur la surface créée. D'un point de vue plus pragmatique, les méthodes de peinture volumique nécessitent un processus de rendu spécifique et posent des problèmes de filtrage notamment. Aussi nous considérons qu'il devient naturel de passer au monde purement surfacique lorsque l'on souhaite décorer la surface. Ceci nous permet donc d'écarter les approches basées sur des textures 3D.

L'utilisation d'une texture pour colorer le maillage peut s'apparenter à l'utilisation de CSS pour spécifier l'apparence du contenu d'un HTML. On souhaite découpler le contenu, ici la forme 3D, de sa visualisation, qui ici est donnée notamment par un choix de couleur. L'approche associant des couleurs aux éléments du maillage semble donc également peu appropriée puisqu'elle nécessite de modifier la tessellation du modèle pour gérer le détail. De plus, la mémoire vidéo des cartes graphiques ne cesse d'évoluer et il est plus facile de sur ou sous échantillonner une image pour s'adapter au hardware graphique plutôt que de modifier un maillage. L'approche par texture est donc plus évolutive. Par ailleurs, l'utilisation de texture permet également à l'artiste des retouches avec des outils dédiés aux traitements 2D. L'utilisation d'images permet un couplage avec des outils artistiques plus traditionnels. Enfin, en utilisant des techniques semblables au *bump mapping*, on peut également facilement ajouter des petits détails là où il faudrait une tessellation très fine pour obtenir un résultat semblable. Les textures permettent donc une expressivité importante pour un coût de rendu relativement faible.

Nous supposons que le paramétrage de la surface est donné en entrée et ne sera pas modifié. Si aucun paramétrage n'est donné, un paramétrage quelconque sera calculé une fois pour toute et ne sera plus modifié. Cette restriction qui peut sembler forte se justifie par le fait que l'utilisateur a pu vouloir créer un paramétrage *ad hoc* et précis pour son modèle et qu'il ne souhaite certainement pas voir celui-ci modifié.



(a) Système ne gérant pas les traits. (b) Système gérant les traits.

FIG. 7.1: Importance de la gestion de traits.

Nous supposerons seulement que le paramétrage est conforme, un choix qui semble raisonnable puisque la plupart des approches de paramétrage présentées dans la littérature fait cette hypothèse. Seule la résolution de la texture sera paramétrable. On s'impose une méthode la plus légère possible pour être compatible avec le plus de matériel possible ; nous ne traiterons donc que la gestion des entrées utilisateur et la mise à jour de la texture à appliquer sur la surface.

Nous reprenons et étendons certains des critères de DeBry *et al.* [DGPR02] souhaitables dans un bon système de peinture de surface :

- peindre uniquement les faces cibles et ce, directement dans la 3D ;
- masquer autant que possible le paramétrage du modèle et donc gérer les frontières de la mise à plat ;
- gérer la notion de trait.

La gestion de trait est importante pour prendre en compte le fait que le système n'enregistre pas des actions continues. C'est notamment primordial dans le cas où le système subirait un fort ralentissement temporaire (fig. 7.1).

Nous allons donc voir comment ajouter des détails de couleur et des détails géométriques sur une surface en respectant ces différents critères.

Dans un premier temps, nous nous intéresserons au problème de l'ajout de couleur sur une surface. Nous analyserons une première approche simple avant de proposer une nouvelle solution de mode d'ajout de couleurs.

Les atlas de texture que nous avons utilisés ont été générés par l'approche LSCM, proposée par Lévy *et al.* [LPRM02], afin de minimiser les distortions et d'utiliser au mieux l'espace texture.

7.2 Critique de la peinture dans l'espace paramétrique

Le fonctionnement d'un système de peinture dans l'espace paramétrique est simple (fig. 7.2). Les entrées écran de l'utilisateur sont traduites en positions 3D

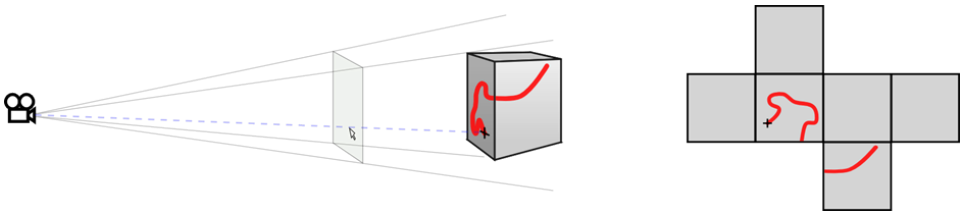


FIG. 7.2: Principe général de la peinture dans l'espace paramétrique.

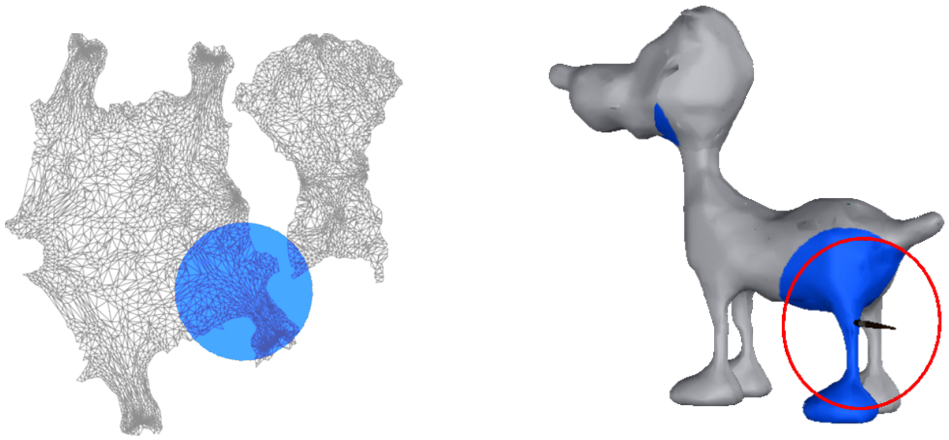


FIG. 7.3: Modification de faces non visibles par peinture dans l'espace paramétrique.

elles mêmes traduites en positions dans l'espace de texture grâce au paramétrage du modèle. Le pinceau est appliqué directement dans l'espace paramétrique et la texture est mise à jour.

Regardons à présent si cette approche, au fonctionnement très simple, permet de répondre aux critères que nous avons fixés précédemment :

- *peindre les faces cibles et ce, directement dans la 3D* : la bijection entre l'espace paramétrique et la surface permet de peindre directement en 3D. Les entrées utilisateurs sont converties en coordonnées sur la surface et on fournit directement un retour à l'utilisateur dans la 3D ; le système masque la conversion des données dans l'espace paramétrique.

  Cependant l'application directe du pinceau dans la texture n'assure pas de peindre uniquement les faces désirées. La configuration du paramétrage n'est pas prise en compte. Aussi il est possible qu'en peignant près du bord du paramétrage, des faces non désirées soient recouvertes par le pinceau produisant un résultat non souhaité (fig. 7.3).

  Ce problème pourrait en partie être levé en comparant les distances des triangles dans la 3D aux distances des triangles dans le paramétrage. Cette solution permettrait de ne peindre que les triangles dont la distance au pinceau est inférieure au rayon du pinceau. Même avec ce correctif, certaines faces non visibles seraient toujours impactées par le pinceau ; il est donc nécessaire de prendre également en compte l'information des faces visibles.

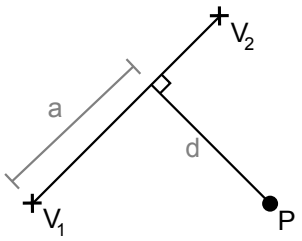


FIG. 7.4: Repérage de la position du pinceau par rapport à une arête bord.

- *masquer autant que possible le paramétrage du modèle et donc gérer les frontières de la mise à plat* : comme nous l'avons vu, le paramétrage d'un modèle non homéomorphe à un disque nécessite une découpe de la surface. Lors du paramétrage, les arêtes situées le long de ce découpage sont ainsi dupliquées. Si le système ne gère pas ces bords, le pinceau risque d'être coupé au niveau des coutures produisant un résultat peu souhaitable (fig. 7.5).

  Une solution à ce problème consiste à repérer toutes les arêtes situées sur le bord qui sont dupliquées ; il s'agit de l'ensemble des arêtes bord[1] dans la 2D qui ne sont pas bord dans la 3D. Lorsque le pinceau touche une arête bord, on peut calculer sa distance à l'arête bord la plus proche

$$a = \overrightarrow{V_1P} \cdot \overrightarrow{V_1V_2} \quad d = \frac{\|\overrightarrow{V_1P} \wedge \overrightarrow{V_1V_2}\|}{2\|\overrightarrow{V_1V_2}\|}.$$

  Pour prendre en compte la présence des bords, il suffit de placer le pinceau à

$$\alpha = \frac{a\|\overrightarrow{V_1'V_2'}\|}{\|\overrightarrow{V_1V_2}\|} \quad \delta = \frac{d\|\overrightarrow{V_1'V_2'}\|}{\|\overrightarrow{V_1V_2}\|}$$

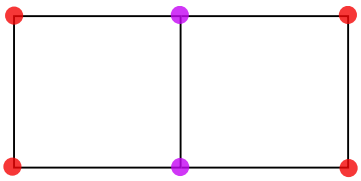
  de l'arête $V_1'V_2'$ duplication de V_1V_2, en prenant en compte l'éventuelle différence de longueur entre les arêtes.

  En pratique les résultats obtenus sont peu satisfaisants (fig. 7.5). Rien n'assure en effet que la déformation du paramétrage soit la même des deux côtés de la couture ce qui peut entraîner de nouveaux artefacts.

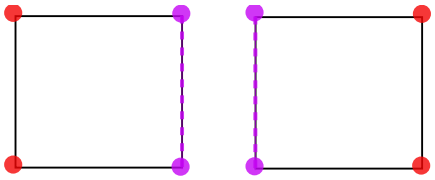
  Piponi et Borshukov [PB00] proposent également d'utiliser une fonction d'interpolation autour des coutures pour lisser le résultat. L'approche donne de bons résultats pour la seule peinture de couleurs. Cependant, dans le cas où on souhaiterait plaquer une image à travers la couture, le lissage introduit produirait un artefact. Le problème de gestion des coutures pour un système de peinture dans l'espace paramétrique ne semble pas avoir de solution générale satisfaisante.

- *gérer la notion de trait* : l'approche naïve consiste à relier les points successifs dans la 2D par un trait. Cette solution est simple à mettre en place mais malheureusement elle peut de nouveau poser des problèmes à proximité des coutures. En effet si une couture est traversée entre deux points enregistrés par le système, il est possible que le segment reliant les deux points dans le monde paramétrique ne corresponde pas au chemin réellement parcouru dans la 3D et introduise un trait non voulu sur le modèle (figure 7.6). Une solution

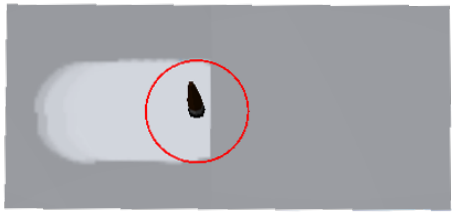
[1] pour rappel, une arête bord est une arête qui ne possède qu'un triangle adjacent



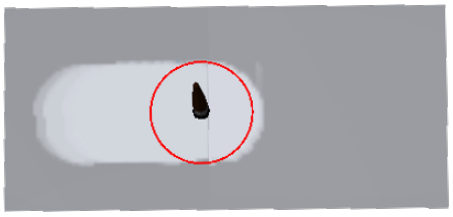
(a) Modèle non paramétré.



(b) Modèle paramétré avec découpe et duplication de l'arête violette.

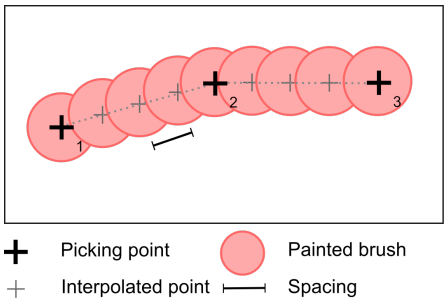


(c) Application du pinceau à proximité de la frontière.



(d) Tentative de correction des problèmes à la frontière.

FIG. 7.5: Problème de gestion des bords pour la peinture dans l'espace paramétrique.



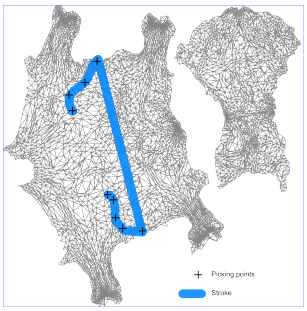


FIG. 7.6: Gestion de trait dans l'espace paramétrique.

pourrait être de calculer le plus court chemin entre les deux points enregistrés mais sur la surface 3D.

Globalement, si l'approche par peinture dans l'espace paramétrique semble intéressante au premier abord, les problèmes liés à la gestion des coutures sont difficilement solvables en toute généralité. Comme nous avons considéré l'utilisation d'un paramétrage conforme, nous n'avons pas considéré les problèmes de distortion gérés par Ray [Ray03] et nous avons simplement pris en compte les facteurs de dilatation. Les solutions possibles pour corriger ces défauts font notamment intervenir des informations liées au point de vue (dans le cas de la peinture sur les faces visibles) ou des données liées à la surface 3D (dans le cas de la gestion de trait) donc une solution purement basée sur l'espace paramétrique n'est pas réalisable si l'on veut remplir les critères que nous nous sommes imposés.

Hanrahan et Haeberli [HH90] suggéraient deux autres modes d'ajout de couleur, à savoir par projection écran ou dans l'espace tangent. De nombreux logiciels fournissent des solutions par projections écran. L'utilisateur peint directement sur

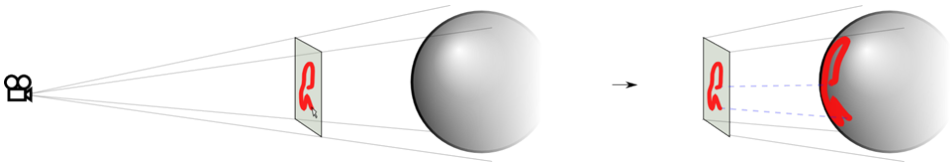


FIG. 7.7: Principe général de la peinture par projection écran.

l'écran et le motif est ensuite projeté sur la surface. Cette solution a notamment l'avantage de gérer intrinsèquement les frontières du paramétrage et ne permet de peindre que les faces visibles. La gestion des traits peut se faire dans l'espace écran [HH90, Elk, IC01]. L'approche a également l'avantage de pouvoir être implémentée très efficacement puisque la projection réelle sur la surface peut n'être effectuée que lors du changement de point de vue.

Cette approche répond donc globalement aux contraintes que nous avons posées sur le système de peinture. Néanmoins, la projection va introduire de fortes distortions à proximité des zones à forte courbure (voir la projection au niveau des pôles sur la figure 7.7). Ceci demande donc un effort à l'utilisateur qui doit corriger ce défaut en modifiant son point de vue. Des tests utilisateur, réalisés au sein de Dassault Systèmes, prouvent que le grand public éprouve des difficultés d'utilisation du contrôle du point de vue.

Nous proposons maintenant une approche pouvant être vue comme un compromis entre la peinture par projection écran et la peinture dans l'espace tangent.

7.3 Peinture par projections locales

La peinture par projection écran peut être vue comme l'utilisation d'une bombe de peinture projetant la couleur sur la surface depuis le point de vue de l'utilisateur. La bombe de peinture est contrainte de se déplacer dans un plan figé. Cette approche permet de répondre aux critères que nous avions fixés mais nécessitent de nombreuses manipulations de point de vue.

Le problème peut être réglé en distinguant la tête de l'artiste,*i.e.* le point de vue à l'écran, de sa main qui controle la bombe. En modélisant le pinceau comme un trièdre de l'espace dissocié de la vue écran, et en reprenant le concept de projection non plus dans le repère caméra mais dans ce repère pinceau, on redonne à l'outil une dimension volumique et une plus grande liberté à l'artiste (fig. 7.8).

Le fonctionnement est globalement le même que pour la peinture par projection écran à ceci près que l'écriture dans la texture est faite au coup par coup puisque le morceau de surface « vu » par le pinceau change en permanence.

Notre implémentation fonctionne en deux étapes :

1. on commence par trouver un paramétrage de la surface dans la vue pinceau (fig. 7.9) ; on détermine donc les triangles visibles et la projection des sommets de ces triangles dans la vue du pinceau.

2. on déforme la texture associée au paramétrage pinceau pour la faire correspondre au paramétrage de la surface (fig. 7.10).

Notre implémentation est accélérée par la carte graphique en utilisant une succession de deux rendus. Le premier rendu sert à déterminer le paramétrage écran

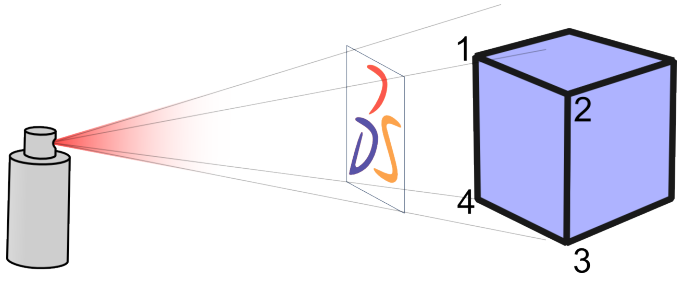


FIG. 7.8: Projection d'une image sur un cube.

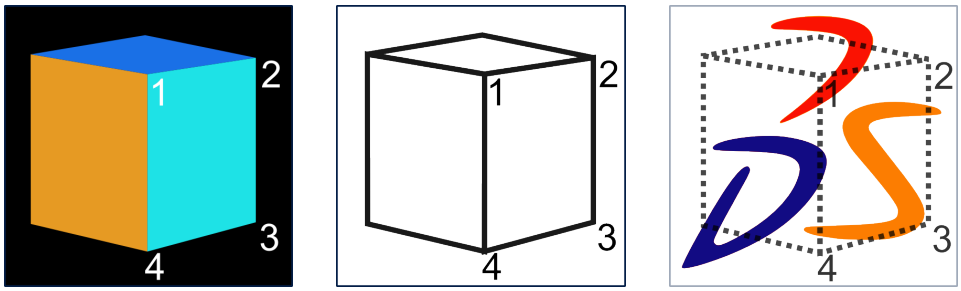


FIG. 7.9: Détermination du paramétrage pinceau de la surface.

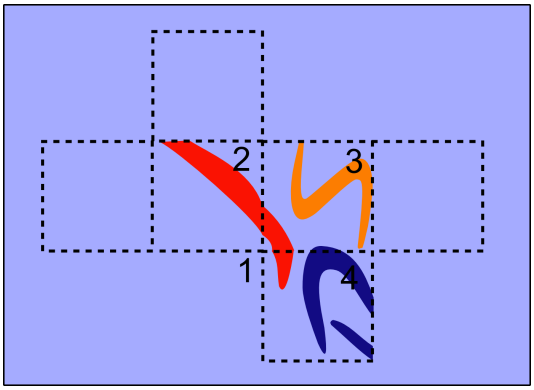


FIG. 7.10: Texture après projection locale.

en effectuant un rendu avec une couleur par face. On effectue ensuite un rendu en projection orthographique en considérant que le paramétrage pinceau est un paramétrage du paramétrage de la surface (qui peut lui même être vu comme un maillage). Le premier rendu utilise des paramètres de projection associés au pinceau. Les deux rendus sont effectués dans la résolution de la texture finale.

Les nombreuses frontières sont bien gérées par notre approche (fig. 7.11). Cependant, un problème d'échantillonnage de la texture par la carte graphique rend les coutures visibles. Pour palier à ce problème, on peut utiliser une diffusion des couleurs dans la texture.

Nous proposons de corriger ce problème par une technique plus simple et pouvant être implémentée efficacement. Afin de faire « baver » les bords de la texture, on ajoute un rendu du paramétrage des arêtes bord avec une épaisseur ; on reprend

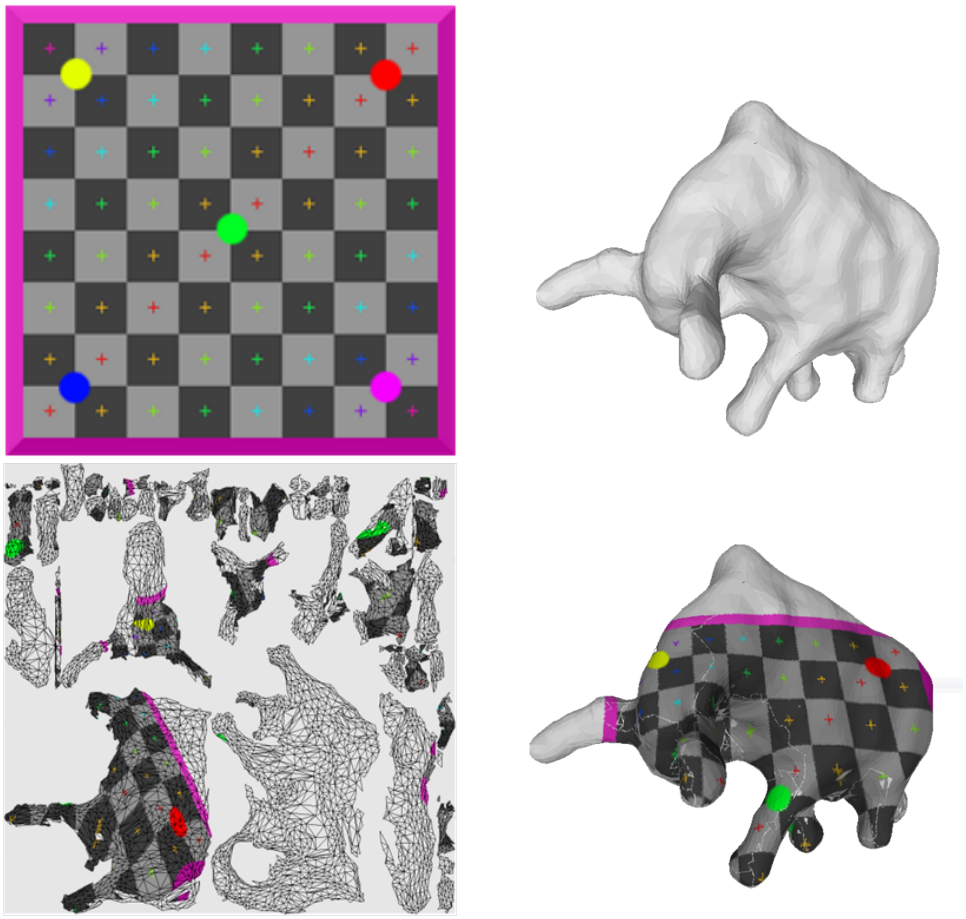


FIG. 7.11: Projection de damier sur un maillage de mammouth.

évidemment les coordonnées du paramétrage pinceau afin qu'elles soient texturées correctement. On effectue ensuite le rendu des triangles dans le même *buffer* de sorte que le rendu des arêtes n'altère pas le rendu final (fig. 7.12).

Notre approche étend les approches de peinture par projection écran en dissociant l'outil de peinture de la vue écran. Un avantage à cette approche est aussi que cette modélisation permet une utilisation directe dans des environnements immersifs [GEL99, KFM+01] alors que les approches reliant trop fortement la vue utilisateur à l'outil (basée sur du *picking* ou projection écran) seraient plus délicates à utiliser.

Cependant, dans le cadre d'une utilisation à la souris, la manipulation d'un trièdre est délicate. Dans ce cas, nous proposons d'effectuer un picking à partir de la position souris et de placer la bombe à une certaine hauteur via la normale à la face « pickée » (fig. 7.13). Le contrôle de la taille du pinceau peut être réalisé de deux façons : soit en modifiant les propriétés de projection lié au pinceau (jouer sur le paramètre de *field of view*) soit en modifiant la hauteur à la surface.

Nous avons globalement repris les paramètres proposés par Ray [Ray03] au niveau de l'interface de gestion de couleur qui propose soit un mode pinceau simple permettant d'ajouter de la couleur avec une forme particulière de pinceau, soit un

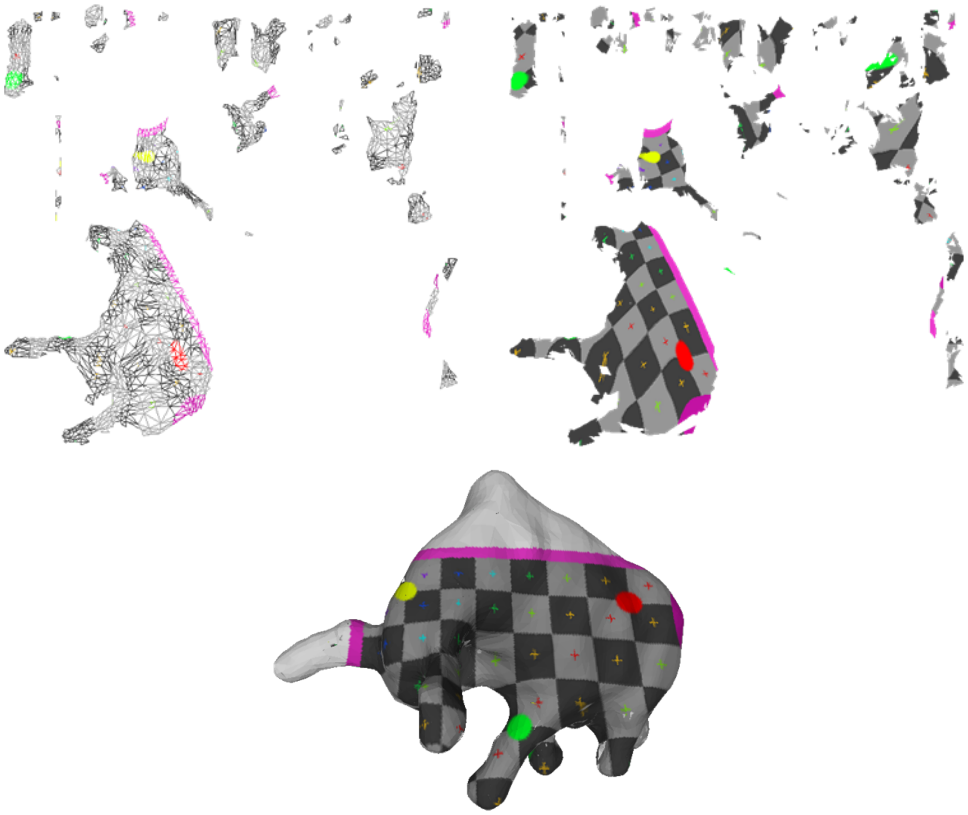


FIG. 7.12: Correction des problèmes d'échantillonnage de texture par rendu des arêtes bord.

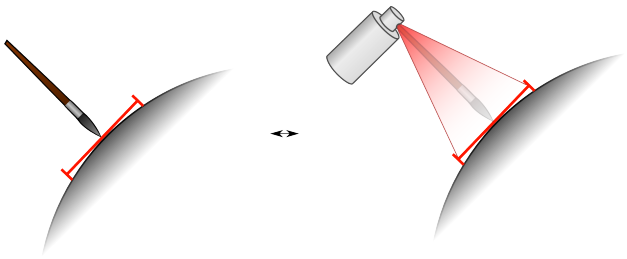


FIG. 7.13: Utilisation de l'approche par projection locale à la souris.

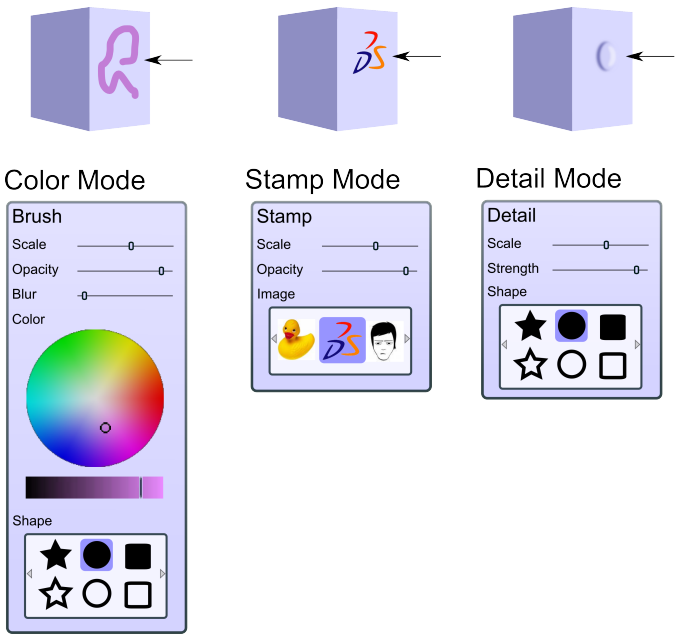


FIG. 7.14: Interface utilisateur de contrôle des outils de peinture.

mode tampon permettant d'appliquer une image existante sur la surface (fig. 7.14). Un troisième mode, présenté dans la section suivante permet également d'ajouter du détail.

Comme il est important que l'utilisateur ait un retour direct aussi bien de l'outil qu'il manipule que du résultat de ses actions, nous affichons un outil correspond au mode courant (pinceau, tampon ou poinçon) directement dans la 3D.

7.4 Ajout de détail géométrique

Dans les processus actuels de création, l'ajout de détail géométrique est usuellement effectué en deux étapes. Le modèle traité est d'abord subdivisé de façon très fine puis sculpté à ce niveau. Une fois fini, le résultat de la sculpture est utilisé pour créer une *normal map* i.e. une texture de normales en chaque point de la surface initiale.

Cette approche présente plusieurs problèmes :

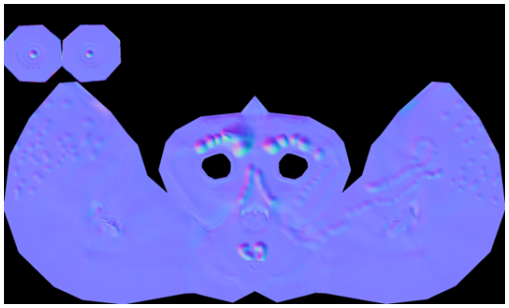
- l'utilisation d'une tessellation très fine nécessite d'importantes ressources matérielles ;
- le résultat final n'est pas exactement le résultat que l'utilisateur avait à l'écran pendant sa création ;
- ce processus demande une certaine compréhension des mécanismes sous-jacents de la part de l'utilisateur.

7.4.1 Sur la création de détail en temps réel

Nous reprenons l'idée présentée par Blinn [Bli78]. Comme nous l'avons vu, l'idée est d'attribuer aux points de la surface une normale qui serait celle qu'ils auraient si chaque point de la surface était déplacé le long de sa normale avec une hauteur

(a) Modèle initial (3800 faces).

(b) Modèle sculpté (685000 faces).



(c) *Normal map* générée.

(d) Modèle initial avec *normal map*.

FIG. 7.15: Processus usuel d'ajouts de détail.

F. De cette façon, la surface n'est pas modifiée mais son éclairage est celui qu'elle aurait si elle était effectivement déformé selon la fonction F en tout point.

Si on note P le point de la surface, N sa normale et F l'élévation de ce point, le nouveau point s'écrit

$$P' = P + F \cdot \frac{N}{||N||}.$$

Par ailleurs, si P_u et P_v sont deux vecteurs tangents à la surface en P on a

$$N = P_u \wedge P_v.$$

On cherche maintenant à évaluer la normale à la surface perturbée en P'. On a par définition

$$\begin{aligned} N' = \frac{\partial P'}{\partial u} \wedge \frac{\partial P'}{\partial v} &= \tfrac{\partial}{\partial u}(P + F \cdot \tfrac{N}{||N||}) \wedge \tfrac{\partial}{\partial v}(P + F \cdot \tfrac{N}{||N||}) \\ &= \left(P_u + F_u \cdot \tfrac{N}{||N||} + F \cdot \tfrac{N_u}{||N||}\right) \wedge \left(P_v + F_v \cdot \tfrac{N}{||N||} + F \cdot \tfrac{N_v}{||N||}\right) \\ &\quad \text{en faisant l'hypothèse que } F \text{ est négligeable,} \\ &\approx \left(P_u + F_u \cdot \tfrac{N}{||N||}\right) \wedge \left(P_v + F_v \cdot \tfrac{N}{||N||}\right) \\ &= N + \tfrac{F_u \cdot (N \wedge P_v) - F_v \cdot (N \wedge P_u)}{||N||}. \end{aligned}$$

$(N \wedge P_u)$ et $(N \wedge P_v)$ sont deux vecteurs situés dans le plan tangent et la normale est donc perturbée par deux vecteurs tangents à la surface et proportionnellement aux dérivées partielles de la fonction de hauteur. Si l'utilisateur peint dans une texture la hauteur de chaque point, il nous suffit donc de calculer le gradient de cette image pour obtenir les perturbations de normales associées et donner l'impression de hauteur sur la surface. Le gradient peut s'évaluer avec le schéma de différences finies centrées (fig. 7.16) *i.e.*

$$\nabla F(u,\, v) \approx \frac{1}{2h} \begin{pmatrix} F(u+h,\, v) - F(u-h,\, v) \\ F(u,\, v+h) - F(u,\, v-h) \end{pmatrix}.$$

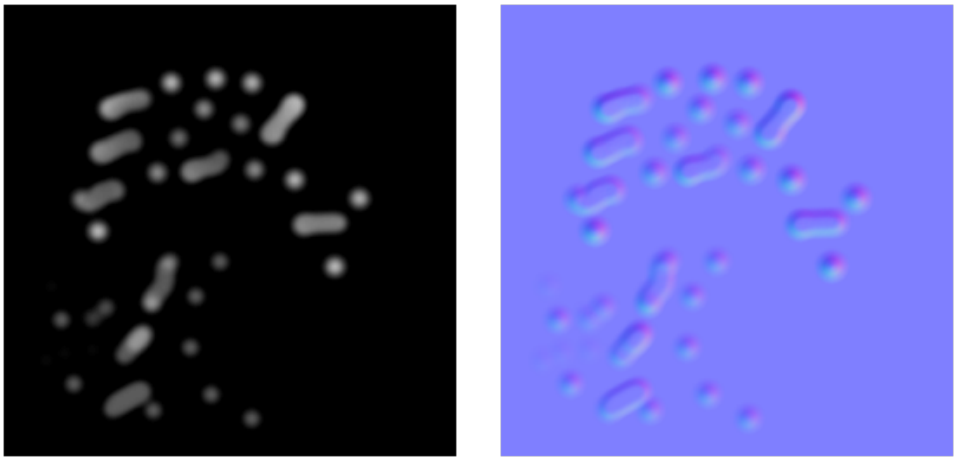


FIG. 7.16: Transformation d'une carte de hauteur en carte de normales.

Les normales sont données dans l'espace tangent ce qui explique que les textures de *normal mapping* soient à dominance bleue (qui dans notre système correspondrait à la couleur de la normale non perturbées).

Si cette approche permet d'obtenir de bons résultats temps réel, elle n'est pas exempte de défauts.

L'approche par *normal map* ne prend pas en compte le point de vue (fig. 7.17). Aussi, selon le point de vue utilisé, le faux détail sera visible en vue de face et ne le sera plus avec un point de vue tangentiel au détail. Si ce n'est pas gênant pour des petits détails, ça le devient pour des bosses plus importantes. Par ailleurs, ces grosses bosses ne produisent pas l'ombre que l'on attendrait. Différentes approches proposent de modifier le rendu pour prendre en compte les fortes courbures créées et produisent des ombres cohérentes [KTI+01, Tat06] ; l'idée est d'utiliser à la fois une *normal map* et une carte de hauteur pour recréer le parallaxe. Ces techniques donnent de très bons résultats mais elles sont liées au moteur de rendu et ne modifient pas réellement la surface. La plupart des moteurs actuels utilisent le *normal mapping* mais pas encore le *parallax mapping*. Le rendu de la surface pourra donc être différent selon le moteur de rendu utilisé. Nous souhaiterions nous affranchir de cette dépendance et proposer un moteur d'ajout de détail modifiant la surface explicitement.

(a) Modèle avec *normal map*

(b) Modèle sculpté

FIG. 7.17: Problème de qualité lié à l'utilisation de *normal map*.

7.4.2 Vers un moteur intelligent d'ajouts de détail

Les processus d'ajout de détail actuels procèdent en ajoutant des hautes fréquences au modèle sculpté, puis, une fois les détails ajoutés, ces hautes fréquences sont reportées dans une *normal map*.

Nous avons vu qu'il était possible de peindre du petit détail efficacement en écrivant dans une carte de hauteur, ou *height map*, générant automatiquement la *normal map* associée.

L'inconvénient de la première approche est qu'elle nécessite temporairement l'utilisation d'une surface très dense tandis que la seconde approche ne permet pas de représenter efficacement les basses et moyennes fréquences, correspondant à des déformations plus importantes de la surface.

Un couplage des deux approches est envisageable pour fournir un moteur d'ajout de détail intelligent. La carte de hauteur peinte pourrait être traitée de façon à séparer les basses fréquences des hautes fréquences (fig. 7.19) :

- les zones ayant du détail basse fréquence pourraient être affinées localement et la géométrie ajoutée déplacée selon la carte des hauteurs ;
- les hautes fréquences pourraient être toujours converties en *normal map*

Notons que le *displacement mapping* est de plus en plus supporté et les basses fréquences pourraient générer une carte de déplacement (fig. 7.18).

L'approche est potentiellement limitante car elle associe un unique nouveau point à chaque point de la surface d'origine. Le défaut peut se corriger en calculant un nou-

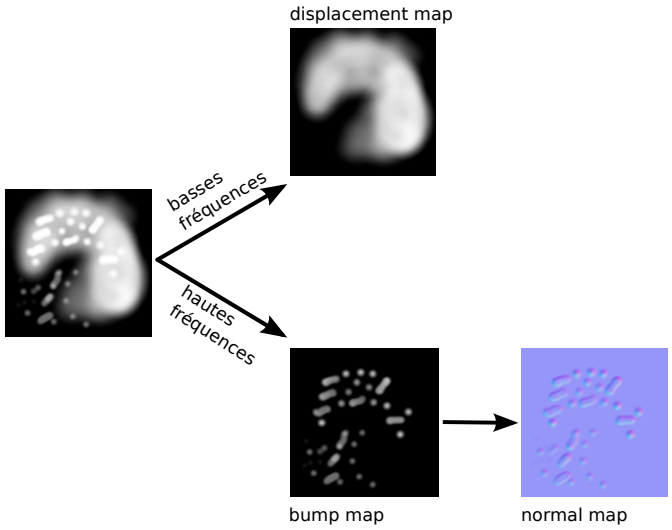


FIG. 7.18: Décomposition des fréquences du détail.

veau paramétrage pour les zones basses fréquences ayant été affinées ; on considère alors la surface détaillée comme nouvelle surface support du détail.

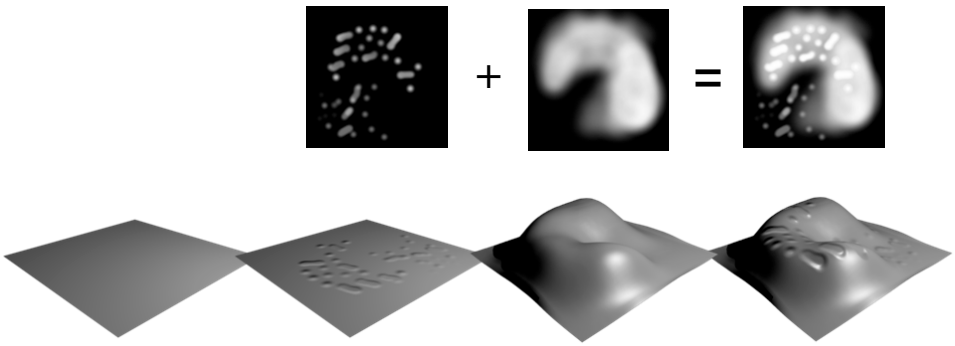


FIG. 7.19: Application de la décomposition du signal de détail.

7.5 Conclusion

Nous avons défini une approche unifiée de peinture de couleurs et de détail géométrique. L'ajout de couleurs est effectué par peinture dans une texture de couleurs et le détail géométrique est peint dans une texture de hauteur dont les hautes et basses fréquences sont gérées différemment.

L'ajout de ces détail se fait au moyen de projections locales qui étendent les approches existantes de peinture par projection écran. Notre approche répond aux critères techniques nécessaires pour que le système soit utilisable sans contrainte d'utilisation et rend l'expérience utilisateur plus aisée. Une utilisation dans un

environnement immersif [KFM+01, BRF01] est directement envisageable puisque l'approche ne repose pas sur une modélisation liée à la visualisation sur écran 2D.

L'approche permettra également de considérer des effets « calligraphiques » en ajoutant la notion de vitesse et d'orientation à la normale dans la notion de trait.

Cette approche par projection locale permet en outre de traiter des modèles paramétrés par face à l'instar des travaux de Burley et Lacewell [BL08]. On peut ainsi utiliser un paramétrage isométrique pour ne pas avoir de distortion. Un autre avantage est que l'approche n'utilise pas la topologie du modèle traité. On peut ainsi travailler sur des soupes de triangles.

Troisième partie

Conclusion

Chapitre 8

Conclusion

Dans ce travail, nous avons cherché à définir un outil, intuitif et non limitatif, permettant la création de formes tridimensionnelles libres.

Bilan

Nous avons d'abord montré l'intérêt d'une approche lagrangienne pour aborder notre problématique. Les systèmes de particules peuvent naturellement modéliser des volumes. De plus, leur topologie est variable et, conceptuellement, chaque particule peut être considérée comme un « petit » élément du matériau manipulé. Les particules peuvent alors être vues comme des *descripteurs de formes*.

Nous avons ensuite défini un cadre de simulation sur systèmes de particules. La méthode *Smoothed Particle Hydrodynamics* permet d'approcher les valeurs de fonctions définies dans l'espace, ou une approximation de valeurs différentielles. La simulation efficace en approche lagrangienne nécessite de pouvoir déterminer rapidement le voisinage d'un point. Nous avons donc mené une étude comparative des structures permettant des recherches spatiales efficaces. Nous avons enfin défini un schéma d'intégration dans le temps.

Sur la base de ces outils, nous avons défini un ensemble de comportements physiques basiques et intuitifs. Ces comportements peuvent être assemblés pour créer de nouveaux matériaux plus riches.

Enfin, les matériaux modélisés étant continus, nous avons cherché à habiller les systèmes de particules par une surface. En reprenant une analogie physique, nous avons utilisé une approche implicite reposant sur la densité de matière. Diverses stratégies, basées sur une reconstruction locale et des stratégies d'échantillonnage, ont été proposées pour optimiser le temps de construction de la surface. Enfin, une implémentation efficace de lissage de la peau a été présentée.

La création de formes peut être facilitée par l'utilisation de modèles existants. Nous avons donc donné des méthodes simples permettant de convertir un modèle surfacique, fermé ou ouvert, en système de particules.

Nous avons ensuite défini un jeu d'outils de manipulation locale de la matière. Ces outils reproduisent les interactions réalisées avec des matériaux de type pâte à modeler dans le monde réel. Divers prototypes, implémentés sur des stations de

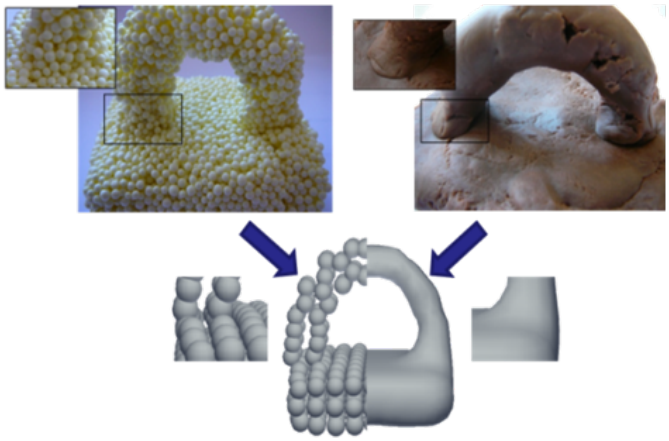


FIG. 8.1: Modèle particulaire et son habillage.

travail standard ou du matériel de réalité virtuelle, ont permis de valider l'intérêt de l'approche lagrangienne pour la création de formes tridimensionnelles libres.

Nous avons enrichi les manipulations de formes possibles en reprenant et proposant des améliorations à une approche de déformation globale.

Enfin, pour donner plus de réalisme aux modèles produits, nous avons présenté une réflexion sur l'ajout unifié de détails géométrique et de couleur.

Notre modélisation synthétise un matériau couplant les avantages des pâtes à modeler lisses et des pâtes à modeler par billes, sans pâtir de leur inconvénients respectifs (fig. 8.1).

Les outils implémentés couvrent une partie importante des besoins du processus complet de création (fig. 3.24). Nous avons cherché des outils suffisamment génériques pour qu'ils ne soient pas limités à notre modélisation par particules et permettent de coupler notre modélisation avec les modélisations usuelles.

L'interopérabilité entre les outils a également été pensée pour fournir la plus grande souplesse dans le processus créatif (fig. 8.2).

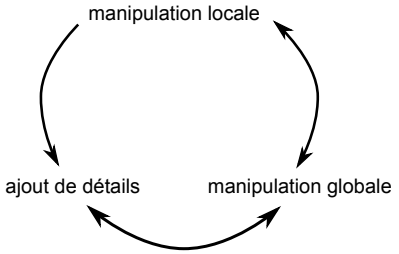


FIG. 8.2: Interopérabilité des outils proposés.

Le passage du mode d'ajout de détail à la manipulation locale serait envisageable en sauvegardant l'ensemble des informations des projections locales. Il sera nécessaire de finir l'implémentation de l'approche d'ajout de détail pour mesurer de la viabilité de ce processus.

La manipulation locale a été expérimentée en contexte immersif. Les outils de dé-

formation globale et d'ajout de détail nécessitent des entrées utilisateur directement compatibles avec les besoins de ce type de matériel. Notre modélisation, destinée, aujourd'hui, aux stations de travail standard, est donc naturellement évolutive vers ces équipements immersifs qui émergent comme une nouvelle norme d'expérience utilisateur.

Notons enfin que deux études récentes vont dans le même sens que nos travaux [Sei09a, Dö09].

Perspectives

Ces travaux constituent un point de départ pour de nombreux développements futurs. L'implémentation réalisée, afin d'être la plus universelle possible, repose sur l'utilisation du *Central Processing Unit* (*CPU*) uniquement. Des premiers tests d'implémentation sur des architectures massivement parallèle, comme celle des *Graphical Processing Unit* (*GPU*), permettent de simuler, en temps réel, des systèmes de taille bien plus importante. Ces tests tendent à montrer que deux ordres de grandeur dans la taille des systèmes manipulés peuvent être gagnés. Adapter les outils à ce genre d'architecture matérielle est donc la première voix développement.

Nous avons cherché à adresser une grande couverture fonctionnelle afin de comparer au mieux notre approche aux approches de modélisation existantes. Les outils implémentés constituent des preuves de concept et nécessitent généralement une étude approfondie pour les rendre plus robustes et permettre leur utilisation par le grand public.

L'outil d'ajout de détail décrit nécessite une implémentation efficace. La décomposition de la carte de hauteurs demande également une réflexion sur le bon niveau de filtrage entre les hautes et basses fréquences.

L'utilisation dans un contexte industriel nécessite également de pouvoir générer des surfaces de haute qualité. Une approche possible serait de remailler la peau obtenue avec un maillage quadrangulaire. La littérature sur le sujet est riche [RLL+06, KNP07], et il semble que les approches reposant sur un paramétrage, nécessitant peu ou pas d'entrée utilisateur, fournissent généralement de bons résultats et constituent une piste de choix.

De nombreuses extensions sont également envisageables. Le modèle de simulation proposé peut être enrichi de nouveaux comportements et supporter des interactions entre plusieurs matériaux pour continuer l'exploration de l'utilisation de paramètres physiques dans le processus de création artistique.

La manipulation et l'assemblage de plusieurs formes peuvent être facilités par l'utilisation d'arbres de type *Blobtree* [SWSJ07].

Des couplages avec d'autres paradigmes de création comme les approches par *sketching*, d'utilisation plus aisée par le grand public [IH03, NISA07], sont possibles.

Notre volonté de fond est la facilitation du processus créatif de formes tridimensionnelles. Il serait intéressant de fournir des outils puissants d'analyse de la forme. Notre outil de sélection va dans ce sens et peut être amélioré. La réflexion pourrait également être poussée pour, par exemple, rechercher automatiquement les symétries d'une forme. Ceci permettrait soit de réduire la taille des systèmes simulés soit d'imposer une symétrie exacte si l'utilisateur a esquissé une symétrie « approchée » [MGP07]. Il est également possible de rechercher des similarités dans le modèle à l'instar de Bokeloh *et al.* [BWS10], afin de créer une grammaires de formes. Ceci permettrait la construction procédurale de systèmes de particules.

Enfin, un couplage des idées présentées dans les différents chapitres serait possible. On peut par exemple chercher à étendre le spectre des énergies pseudo-

physiques utilisées pour la déformation globale. On pourrait également utiliser l'idée de l'ajout de détail par projections locales avec une simulation de fluide visqueux. La « bombe de peinture » modélisée par un trièdre peut servir de source de particules de peinture ; chaque particule, selon sa trajectoire sur la surface, pourrait modifier localement les textures de couleurs ou de hauteur, et les problèmes de continuité aux bords du paramétrage seraient gérés grâce à la continuité intrinsèque des comportements physiques.

Quatrième partie

Annexes

Annexe **A**

Définition des configurations de patchs

Avant de déterminer l'espace mémoire nécessaire pour un patch, nous illustrons chacune des configurations de patch.

On donne à présent les différentes formules permettant de dénombrer les $N_{T,adj}$ triangles adjacents aux triangles réguliers, étant donné les valences des trois sommets de base v_0, v_1 et v_2, la configuration du *patch* (fig. A.1 et A.2) et le niveau de subdivision n.

- E_0V_k $(k = 0 \dots 3)$

$$N_{T,adj} = \sum_{i=0}^{2} |v_i| + 3 \cdot 2^{n+1} - 12 - k$$

- E_1V_k $(k = 2, 3)$

$$N_{T,adj} = \sum_{i=0}^{2} |v_i| + 2 \cdot 2^{n+1} - 9 - k$$

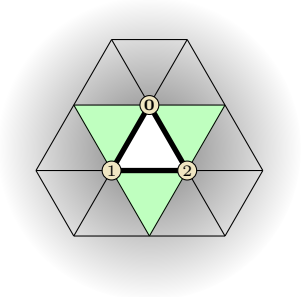
- E_2V_3

$$\begin{aligned} N_{T,adj} &= \sum_{i=0}^{1} |v_i| + 2^{n+1} - 7 \\ &= \sum_{i=0}^{2} |v_i| + 2^{n+1} - 9 \end{aligned}$$

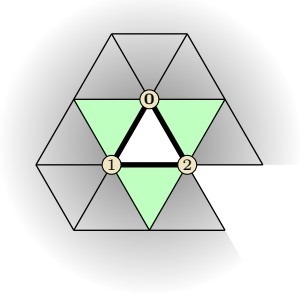
- E_3V_3

$$N_{T,adj} = 0$$

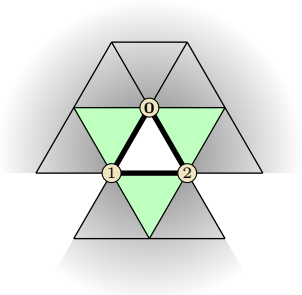
On peut majorer toutes les expressions par $N_{T,adj,max} = 3 \cdot (V_{max} + 2^{n+1}) - 12$.



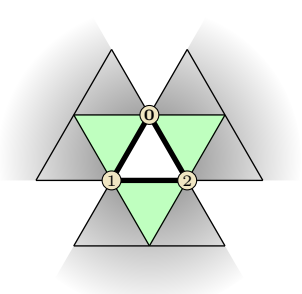
Configuration E_0V_0 (ou standard) :
- 0 sommet ouvert
- 0 arête libre.



Configuration E_0V_1 :
- 1 sommet ouvert
- 0 arête libre.

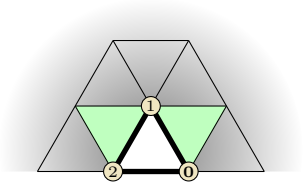


Configuration E_0V_2 :
- 2 sommets ouverts
- 0 arête libre.

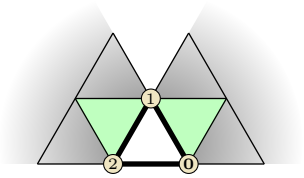


Configuration E_0V_3 :
- 3 sommets ouverts
- 0 arête libre.

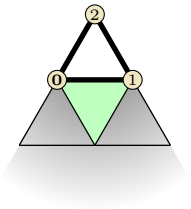
FIG. A.1: Configurations de *patch* de référence (1/2)



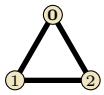
Configuration E_1V_2 :
- 2 sommets ouverts
- 1 arête libre.



Configuration E_1V_3 :
- 3 sommets ouverts
- 1 arête libre.

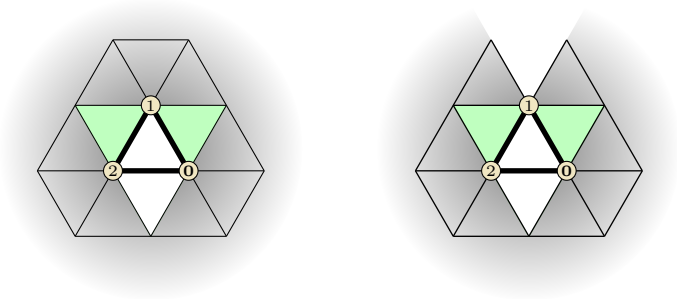


Configuration E_2V_3 :
- 3 sommets ouverts
- 2 arêtes libres.



Configuration E_3V_3 :
- 3 sommets ouverts
- 3 arêtes libres.

FIG. A.2: Configurations de *patch* de réfénce (2/2)



E_1V_2 E_1V_3

FIG. A.3: Cas particuliers des configurations E_1V_2 et E_1V_3

FIG. A.4: Illustration des cas limite (*patch* standard)

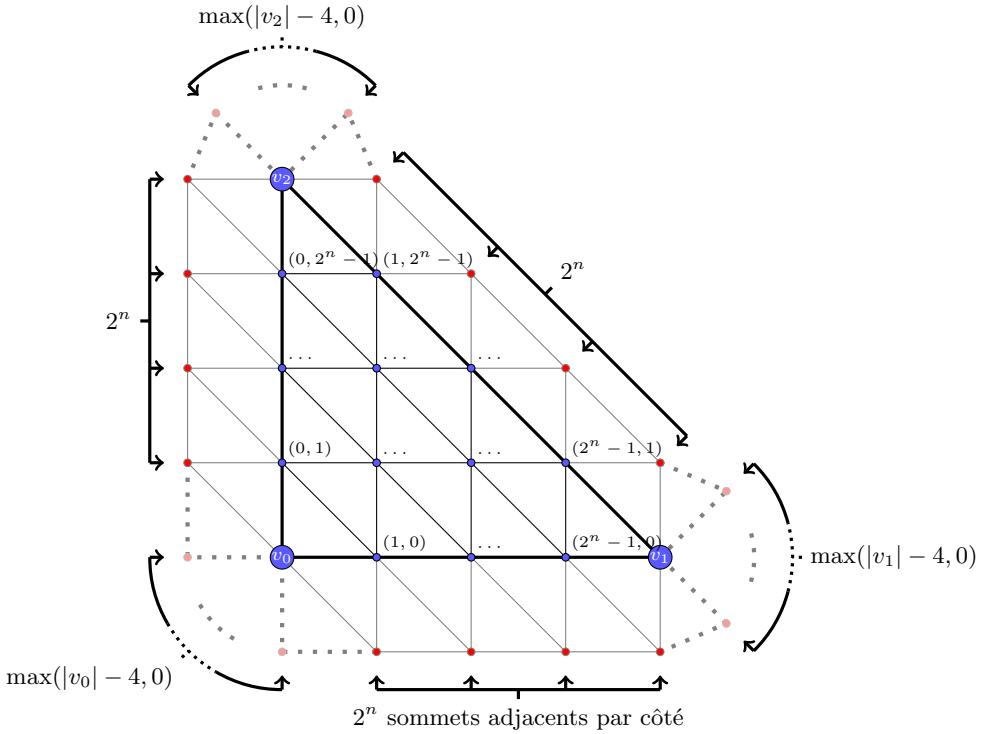


FIG. A.5: Dénombrement des sommets adjacents (*patch* standard)

Annexe **B**

Quelques détails de calculs pour la déformation globale

Toute rotation peut se décomposer comme composition de rotations $\theta_x, \theta_y, \theta_z$ autour des axes canoniques soit, en considérant des rotations d'angles petits et une approximation du premier ordre :

$$\begin{aligned}
R &= \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta_y\cos\theta_z & \sin\theta_x\sin\theta_y\sin\theta_z - \cos\theta_x\sin\theta_z & \cos\theta_x\sin\theta_y\cos\theta_z - \sin\theta_x\sin\theta_z \\ \cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z + \sin\theta_x\sin\theta_y\sin\theta_z & \cos\theta_x\sin\theta_y\sin\theta_z - \sin\theta_x\cos\theta_z \\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{pmatrix} \\
&\underset{(0,0,0)}{\approx} \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{Id} + \begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \end{pmatrix} \wedge \end{pmatrix}
\end{aligned}$$

Une fois la linéarisation de la transformation déterminée, il faut reprojeter la transformation sur l'espace des transformations rigides. La partie délicate est l'obtention d'une rotation qui soit effectivement une matrice orthonormale et non une approximation. Nous détaillons le processus de projection issu des résultats de Horn [Hor87].

On note p_i l'ensemble des 8 sommets modélisant un voxel (parallélépipède rectangle), O le centre de ce voxel. On note

$$P_i = p_i - O = p_i - \frac{1}{n}\sum_i p_i,$$

l'expression des sommets dans le référentiel voxel. On remarque

$$\sum_i P_i = \sum_{i=1}^{n} p_i - O = -n \cdot O + \sum_{i=1}^{n} p_i = 0_{\mathbb{R}^3}.$$

Les points p'_i sont les images des points p_i par une transformation affine

$$p'_i = p_i \wedge \omega_i + \mathrm{v}_i = Ap_i + \mathrm{v}_i.$$

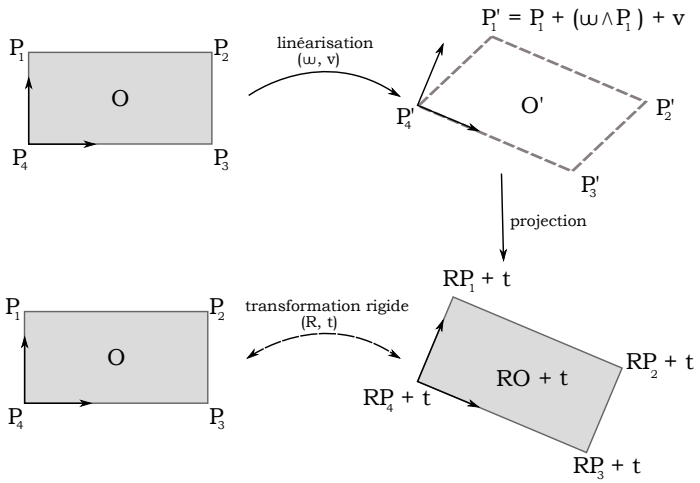


FIG. B.1: Processus de projection rigide.

On exprime également cet ensemble de points dans un référentiel local

$$P'_i = p'_i - O'.$$

On souhaite à présent trouver une transformation rigide, de la forme

$$\mathrm{T(x)} = R\mathrm{x} + \mathrm{t},$$

avec R une matrice de rotation et t un vecteur de translation, permettant de passer « au mieux » de l'ensemble des $\{p_i\}$ aux $\{p'_i\}$ (fig. B.1). On exprime ce problème sous la forme d'un problème de minimisation. On souhaite minimiser l'erreur

$$\begin{aligned}\epsilon &= \textstyle\sum_i ||p'_i - \big(Rp_i + t\big)||^2 \\ &= \textstyle\sum_i ||(P'_i + O') - \big(R(P_i + O) + t\big)||^2 \\ &= \textstyle\sum_i ||P'_i - RP_i - \underbrace{\big(t - O' + RO\big)}_{t_O}||^2\end{aligned}$$

En développant le carré, on a

$$\epsilon = n||t_O||^2 + \textstyle\sum_i 2\langle P'_i - RP_i, t_O\rangle + \sum_i ||P'_i - RP_i||^2,$$

or, par linéarité du produit scalaire, il vient

$$\begin{aligned}\textstyle\sum_i 2\langle P'_i - RP_i, t_O\rangle &= 2\langle\textstyle\sum_i P'_i, t_O\rangle - 2\langle\sum_i RP_i, t_O\rangle \\ &= 2\langle\underbrace{\textstyle\sum_i P'_i}_{=0_{\mathbb{R}^3}}, t_O\rangle - 2\langle R\underbrace{\big(\textstyle\sum_i P_i\big)}_{=0_{\mathbb{R}^3}}, t_O\rangle \\ &= 0.\end{aligned}$$

Le problème se ramene donc à la minimisation de

$$\epsilon = n||t_O||^2 + \sum_i ||P'_i - RP_i||^2.$$

Le premier terme est positif ou nul donc le minimum de l'erreur est rencontré pour

$$||t_O|| = 0 \Leftrightarrow t_O = t - O' + RO = 0_{\mathbb{R}^3} \Leftrightarrow t = O' - RO.$$

Par conséquent, une fois la rotation déterminée, la translation s'obtient directement.

Il reste à présent à minimiser

$$\begin{aligned}\epsilon &= \sum_i ||P'_i - RP_i||^2 \\ &= \sum_i \left(||P'_i||^2 - 2\langle P'_i, RP_i\rangle + ||RP_i||^2\right) \\ &= \underbrace{\sum_i \left(||P'_i|| - ||RP_i||\right)^2}_{\geq 0} + 2\Bigg(\underbrace{\langle ||P'_i||^2, ||RP_i||^2\rangle}_{\geq 0} - \langle P'_i, RP_i\rangle\Bigg)\end{aligned}$$

et on voit que minimiser ϵ' revient à maximiser $\epsilon' = \sum_i \langle P'_i, RP_i\rangle$.

Horn propose d'utiliser une représentation de la rotation par des quaternions. On introduit les quaternions unitaire $\dot{q} = a + \mathbf{i}b + \mathbf{j}c + \mathbf{k}d$ représentant la matrice R et les quaternions imaginaires purs $\dot{p}_i = \mathbf{i}p_{i,x} + \mathbf{j}p_{i,y} + \mathbf{k}p_{i,z}$ et $\dot{p}'_i = \mathbf{i}p'_{i,x} + \mathbf{j}p'_{i,y} + \mathbf{k}p'_{i,z}$ représentant respectivement les vecteurs P_i et P'_i.

On peut réécrire ϵ' sous la forme

$$\begin{aligned}\epsilon' = \sum_i \langle P'_i, RP_i\rangle &= \sum_i \langle \dot{p}'_i, \dot{q}\dot{p}_i\dot{q}^*\rangle \\ &= \sum_i \langle \dot{P'}_i\dot{q}, \dot{P}_i\dot{q}\rangle \\ &= \sum_i \dot{q}^t \dot{P'}_i^t \dot{P}_i \dot{q} \\ &= \dot{q}^t \underbrace{\left(\sum_i \dot{P'}_i^t \dot{P}_i\right)}_{M} \dot{q}\end{aligned}$$

où l'on a exprimé le produit de quaternion sous forme de produit matrice-vecteur. Les expressions de ces vecteurs sont, en prenant en compte le sens de la multiplication, à droite ou à gauche, et le fait que les quaternions représentant les vecteurs sont imaginaires purs, on a

$$\dot{P}'_i = \begin{pmatrix} 0 & -p'_{i,x} & -p'_{i,y} & -p'_{i,z} \\ p'_{i,x} & 0 & p'_{i,z} & -p'_{i,y} \\ p'_{i,y} & -p'_{i,z} & 0 & p'_{i,x} \\ p'_{i,z} & p'_{i,y} & -p'_{i,x} & 0 \end{pmatrix},$$

et

$$\dot{P}_i = \begin{pmatrix} 0 & -p_{i,x} & -p_{i,y} & -p_{i,z} \\ p_{i,x} & 0 & p_{i,z} & -p_{i,y} \\ p_{i,y} & -p_{i,z} & 0 & p_{i,x} \\ p_{i,z} & p_{i,y} & -p_{i,x} & 0 \end{pmatrix}.$$

En développant, on trouve que le produit $\dot{P}'^t_i \dot{P}_i$ est une matrice symétrique réelle ; par conséquent, $M = \left(\sum_i \dot{P}'^t_i \dot{P}_i\right)$ est également une matrice symétrique réelle et est donc diagonalisable et à valeurs propres réelles.

Notons $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ les 4 valeurs propres de M et (e_1, e_2, e_3, e_4) leurs vecteurs propres associés formant une base orthonormale.
En décomposant $\dot{q} = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3 + \alpha_4 e_4$, et en notant λ_m la plus grande valeur propre, il vient

$$\epsilon' = \dot{q}^t M \dot{q} = \alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \alpha_3^2\lambda_3 + \alpha_4^2\lambda_4 \leq \lambda_m(\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2) = \lambda_m,$$

puisque le quaternion $\dot{q}$ est unitaire (et $\alpha_1^2 + \alpha_2^2 + \alpha_3^2 + \alpha_4^2 = \langle \dot{q}, \dot{q}\rangle$).

La rotation recherchée nécessite donc la détermination du vecteur propre associé à la plus grande valeur propre de la matrice M. Le vecteur de dimension 4 peut être interprété comme un quaternion et fournit l'angle et l'axe de rotation. La transformation rigide est ainsi intégralement déterminée.

Bibliographie

[APKG07] B. Adams, M. Pauly, R. Keiser et L. J. Guibas : Adaptively sampled particle fluids. *Dans Proceedings of ACM SIGGRAPH, papers*, 2007.

[ABL95] M. Agrawala, A. C. Beers et M. Levoy : 3D painting on scanned surfaces. *Dans Proceedings of the Symposium on Interactive 3D Graphics*, pages 145–150, New York, NY, USA, 1995.

[AAH+07] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Kornberger, M. Peternell et H. Pottmann : Approximating boundary-triangulated objects with balls. *Dans Proc. 23rd European Workshop on Computational Geometry*, pages 130–133. TU Graz, 2007.

[AM02] T. Akenine-Möller : Fast 3D triangle-box overlap testing. *Journal of Graphics Tools*, 6(1):29–33, 2002.

[AM93] S. Arya et D. M. Mount : Algorithms for fast vector quantization. *Dans* J. A. Storer et M. Cohn, éditeurs : *IEEE Data Compression Conference*, pages 381–390, Snowbird, UT, USA, 1993.

[Bær01] J. A. Bærentzen : On the implementation of fast marching methods for 3D lattices. Rapport technique, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2001.

[BCN03] Y. Bando, B.-Y. Chen et T. Nishita : Animating hair with loosely connected particles. *Computer Graphics Forum*, 22(3):411–418, 2003. Proceedings of Eurographics 2003.

[BRF01] D. Bandyopadhyay, R. Raskar et H. Fuchs : Dynamic shader lamps : Painting on movable objects. *Dans Proceedings of International Symposium on Augmented Reality*, pages 207, Washington, DC, USA, 2001. IEEE and ACM.

[BW98] D. Baraff et A. Witkin : Large steps in cloth simulation. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 43–54, New York, NY, USA, 1998.

[Bar05] L. Barthe : Box-spline et surfaces de subdivision. *Dans AFIG*, 2005.

[Bat67] G. K. Batchelor : *An introduction to fluid dynamics*. Cambridge University Press, 1967.

[Bec09] M. Becker : *Particle-based animation*. Thèse de doctorat, Albert-Ludwigs-Universität Freiburg, 2009.

[BT07] M. Becker et M. Teschner : Weakly compressible sph for free surface flows. *Dans Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 209–217, Aire-la-Ville, Switzerland, Switzerland, 2007.

[BYM05] N. Bell, Y. Yu et P. J. Mucha : Particle-based simulation of granular materials. *Dans Proceedings of ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 77–86, New York, NY, USA, 2005.

[BD02] D. Benson et J. Davis : Octree textures. *ACM Trans. Graph.*, 21(3) :785–790, 2002.

[Ben90a] J. L. Bentley : K-d trees for semidynamic point sets. *Dans Proceedings of ACM SIGGRAPH*, pages 187–197, New York, NY, USA, 1990a.

[Ben90b] W. Benz : Smooth particle hydrodynamics - a review. *Dans* J. R. Buchler, éditeur : *Numerical Modelling of Nonlinear Stellar Pulsations Problems and Prospects*, 1990b.

[Ber09] D. Berkemeyer : Car clay modelling tutorials, 2009. URL http://kolb.wordpress.com/category/car-clay-modelling-tutorials/.

[Bli78] J. F. Blinn : Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph.*, 12(3):286–292, 1978.

[Bli82] J. F. Blinn : A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, 1982.

[BN76] J. F. Blinn et M. E. Newell : Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, 1976.

[Blo98] J. Bloomenthal : Implicit surfaces. *SIGGRAPH Comput. Graph.*, 32 (4):41, 1998.

[BWS10] M. Bokeloh, M. Wand et H.-P. Seidel : A connection between partial symmetry and inverse procedural modeling. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 1–10, New York, NY, USA, 2010.

[BK04] M. Botsch et L. Kobbelt : An intuitive framework for real-time freeform modeling. *Dans Proceedings of ACM SIGGRAPH*, pages 630–634, New York, NY, USA, 2004.

[BPGK06] M. Botsch, M. Pauly, M. Gross et L. Kobbelt : Primo : coupled prisms for intuitive surface modeling. *Dans Proceedings of Eurographics Symposium on Geometry Processing*, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006.

[BPK+07] M. Botsch, M. Pauly, L. Kobbelt, P. Alliez, B. Lévy, S. Bischoff et C. Rössl : Geometric modeling based on polygonal meshes. *Dans Proceedings of ACM SIGGRAPH, courses*, pages 1, New York, NY, USA, 2007.

[BPWG07] M. Botsch, M. Pauly, M. Wicke et M. H. Gross : Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3) :339–347, 2007.

[BRS04] T. Boubekeur, P. Reuter et C. Schlick : Reconstruction locale et visualisation de surfaces de points a l'aide de surfaces de subdivision. *Dans Journées de l'Association Française d'Informatique Graphique*, 2004.

[Bri02] D. Brickhill : Practical implementation techniques for multi-resolution subdivision surfaces. Rapport technique, The Golden Gate Game Company, 2002.

[BL08] B. Burley et D. Lacewell : Ptex : Per-face texture mapping for production rendering. *Dans Proceedings of Eurographics Symposium on Rendering*, 2008.

[CA06] M.-P. Cani et A. Angelidis : Towards virtual clay. *Dans Proceedings of ACM SIGGRAPH, courses*, pages 67–83, New York, NY, USA, 2006.

[CH04] N. A. Carr et J. C. Hart : Painting detail. *Dans Proceedings of ACM SIGGRAPH*, pages 845–852, New York, NY, USA, 2004.

[CBP05] S. Clavet, P. Beaudoin et P. Poulin : Particle-based viscoelastic fluid simulation. *Symposium on Computer Animation 2005*, pages 219–228, 2005.

[CLL+88] H. E. Cline, W. E. Lorensen, S. Ludke, C. R. Crawford et B. C. Teeter : Two algorithms for the three-dimensional reconstruction of tomograms. *Medical Physics*, 15(3):320–327, 1988.

[Coo84] R. L. Cook : Shade trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, 1984.

[Coq90] S. Coquillart : Extended free-form deformation : a sculpturing tool for 3D geometric modeling. *Dans Proceedings of ACM SIGGRAPH*, pages 187–196, New York, NY, USA, 1990.

[CFMU04] S. Cristiano, M. Fiorentino, G. Monno et A. E. Uva : Real-time particle based virtual sculpturing. *Dans ADMAIAS conference*, 2004.

[CvG07] M. A. Cuendet et W. F. van Gunsteren : On the calculation of velocity-dependent properties in molecular dynamics simulations using the leapfrog integration algorithm. *J Chem Phys*, 127:184102, 2007.

[Dasa] Dassault Systèmes : 3DVIA, a. URL http://www.3dvia.com.

[Dasb] Dassault Systèmes : 3DVIA Shape, b. URL http://www.3dvia.com/shapeit.

[Dasc] Dassault Systèmes : 3DVIA Studio, c. URL http://www.3dvia.com/studio/.

[Dasd] Dassault Systèmes : 3DVIA Virtools, d. URL http://www.3ds.com/products/3dvia/3dvia-virtools/.

[DGPR02] D. DeBry, J. Gibbs, D. D. Petty et N. Robins : Painting and rendering textures on unparameterized models. *Dans Proceedings of ACM SIGGRAPH*, pages 763–768, New York, NY, USA, 2002.

[Des97] M. Desbrun : *Modélisation et Animation de Matériaux Hautement Déformables en Synthèse d'Images*. Thèse de doctorat, Institut National Polytechnique de Grenoble, 1997.

[DC96] M. Desbrun et M.-P. Cani : Smoothed particle : A new paradigm for animating highly deformable bodies. *Eurographics Workshop on Computer Anmation and Simulation*, pages 61–76, 1996.

[DC99] M. Desbrun et M.-P. Cani : Space-time adaptive simulation of highly deformable substances. Rapport de recherche 3829, INRIA, 1999.

[DD06] F. Destelle et B. Derdouri : Reconstruction de nuages de points par surface de subdivision. *Dans Journées de l'Association Francophone d'Informatique Graphique*, 2006.

[DC04a] G. Dewaele et M.-P. Cani : Interactive global and local deformations for virtual clay. *Graph. Models*, 66:352–369, 2004a.

[DC04b] G. Dewaele et M.-P. Cani : Virtual clay for direct hand manipulation. *Dans Eurographics*, 2004b.

[Din05] B. M. Dingle : Obtaining fuzzy representations of 3D objects. Rapport technique, Texas A&M University, 2005.

[Din07] B. M. Dingle : *Volumetric Particle Modeling*. Thèse de doctorat, Texas A&M University, 2007.

[Dö09] U. Dönni : Governed design : a new paradigm to shape finding. Mémoire de D.E.A., ETH Zurich, Department of Computer Science, 2009.

[Elk] G. Elkoura : A 3D interactive texture painter.

[FCG00] E. Ferley, M.-P. Cani et J.-D. Gascuel : Practical volumetric sculpting. *Visual Comput.*, 16(8):469–480, 2000. A preliminary version of this paper appeared in Implicit Surfaces'99, Bordeaux, France, sept 1999.

[GH91] T. A. Galyean et J. F. Hughes : Sculpting : An interactive volumetric modeling technique. *Computer Graphics*, 25:267–274, 1991. Proceedings of SIGGRAPH 91.

[GH97] M. Garland et P. S. Heckbert : Surface simplification using quadric error metrics. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 209–216, New York, NY, USA, 1997.

[GM77] R. A. Gingold et J. J. Monaghan : Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[Gooa] Google : Google 3D Warehouse, a. URL http://sketchup.google.com/3dwarehouse.

[Goob] Google : Google Sketchup, b. URL http://sketchup.google.com/.

[Gre10] S. Green : Screen space fluid rendering for games. *Dans Proceedings of Game Developer Conference*, 2010.

[GEL99] A. D. Gregory, S. A. Ehmann et M. C. Lin : intouch : Interactive multiresolution modeling and 3D painting with a haptic interface. *Dans Proceedings of IEEE VR Conference*, pages 45–52, 1999.

[GHDS03] E. Grinspun, A. N. Hirani, M. Desbrun et P. Schröder : Discrete shells. *Dans Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. ACM.

[GCS99] L. Grisoni, B. Crespin et C. Schlick : Multiresolution implicit representation of 3D objects. Rapport technique, University of Bordeaux I, 1999.

[GH95] A. Guéziec et R. Hummel : Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1:328–342, 1995.

[HH90] P. Hanrahan et P. Haeberli : Direct WYSIWYG painting and texturing on 3D shapes. *SIGGRAPH Comput. Graph.*, 24(4):215–223, 1990.

[HKK07] T. Harada, S. Koshizuka et Y. Kawaguchi : Smoothed Particle Hydrodynamics on GPUs. *Dans Computer Graphics International*, pages 63–70, 2007.

[HHS06] V. Havran, R. Herzog et H.-P. Seidel : On the fast construction of spatial data structures for ray tracing. *Dans Proceedings of IEEE Symposium on Interactive Ray Tracing*, pages 71–80, 2006.

[HRE+08] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard et K. van Prooijen : Real-time motion retargeting to highly varied user-created morphologies. *Dans Proceedings of ACM SIGGRAPH*, pages 1–11, New York, NY, USA, 2008.

[HWC+05] C.-C. Ho, F.-C. Wu, B.-Y. Chen, Y.-Y. Chuang et M. Ouhyoung : Cubical marching squares : Adaptive feature preserving surface extraction from volume data. *Dans Proceedings of Eurographics*, volume 24, pages 537–545, 2005.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald et W. Stuetzle : Surface reconstruction from unorganized points. *Dans Proceedings of ACM SIGGRAPH, papers*, 1992.

[Hor87] B. K. P. Horn : Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[HHK92] W. M. Hsu, J. F. Hughes et H. Kaufman : Direct manipulation of free-form deformations. *Dans Proceedings of ACM SIGGRAPH*, pages 177–184, New York, NY, USA, 1992.

[IC01] T. Igarashi et D. Cosgrove : Adaptive unwrapping for interactive texture painting. *Dans Proceedings of the Symposium on Interactive 3D Graphics*, pages 209–216, New York, NY, USA, 2001.

[IH03] T. Igarashi et J. F. Hughes : Smooth meshes for sketch-based free-form modeling. *Dans Proceedings of the Symposium on Interactive 3D graphics*, pages 139–142, New York, NY, USA, 2003.

[Jak03] T. Jakobsen : Gamasutra-features : Advanced character physics, 2003. URL http://www.gamasutra.com/resource_guide/20030121/jacobson_01.shtml.

[Jen97] B. Jenkins : A hash function for hash table lookup, 1997. URL http://www.burtleburtle.net/bob/hash/doobs.html.

[JLW05] S. Jin, R. R. Lewis et D. West : A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21:71–82, 2005.

[JMY+07] A. Jones, I. McDowall, H. Yamada, M. Bolas et P. Debevec : Rendering for an interactive 360° light field display. *Dans Proceedings of ACM SIGGRAPH, papers*, 2007.

[JLSW02] T. Ju, F. Losasso, S. Schaefer et J. Warren : Dual contouring of hermite data. *Dans Proceedings of ACM SIGGRAPH, papers*, 2002.

[KTI+01] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda et S. Tachi : Detailed shape representation with parallax mapping. *Dans Proceedings of ICAT*, pages 205–208, 2001.

[KFM+01] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw et J. J. LaViola, Jr. : Cavepainting : a fully immersive 3D artistic medium and interactive experience. *Dans Proceedings of the Symposium on Interactive 3D Graphics*, pages 85–93, New York, NY, USA, 2001.

[Kei06] R. Keiser : *Meshless Lagrangian methods for physics-based animations of solids and fluids*. Thèse de doctorat, Department of Computer Science, ETH Zürich, 2006.

[KSW04] P. Kipfer, M. Segal et R. Westermann : Uberflow : a gpu-based particle engine. *Dans Proceedings of ACM SIGGRAPH, sketches*. ACM, 2004.

[KW06] P. Kipfer et R. Westermann : Realistic and interactive simulation of rivers. *Dans Proceedings of Graphics Interface*, pages 41–48, Toronto, Ont., Canada, Canada, 2006.

[KNP07] F. Kälberer, M. Nieser et K. Polthier : Quadcover - surface parameterization using branched coverings. *Dans Proceedings of Eurographics*, 2007.

[KB04] L. Kobbelt et M. Botsch : A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28:801–814, 2004.

[KBKv09] P. Krištof, B. Beneš, J. Křivánek et O. Šťava : Hydraulic erosion using Smoothed Particle Hydrodynamics. *Computer Graphics Forum*, 28(2), 2009.

[Kro10] y. E. Krog : GPU-based real-time snow avalanche simulations. Mémoire de D.E.A., Norwegian Univesity of Science and Technology, Department of Computer and Information Science, 2010.

[LHMG02] U. Labsik, K. Hormann, M. Meister et G. Greiner : Hierarchical iso-surface extraction. *J. Comput. Inf. Sci. Eng.*, 2(4):323–329, 2002.

[Lag95] P. Laguna : Smoothed particle interpolation. *Astrophys.J.*, 439 :814–821, 1995.

[LT00] P. Lascaux et R. Théodor : *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, volume 1. Dunod, 2000.

[LH93] J. Lee et H. Herrmann : Angle of repose and angle of marginal stability : molecular dynamics of granular particles. *Journal of Physics A : Mathematical and General*, 26:373–383, 1993.

[LH06] S. Lefebvre et H. Hoppe : Perfect spatial hashing. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 579–588, New York, NY, USA, 2006.

[Len09] T. Lenaerts : *Unified Particle Simulation and Interactions in Computer Animation*. Thèse de doctorat, Katholieke Universiteit Leuven, 2009.

[LLVT03] T. Lewiner, H. Lopes, A. W. Vieira et G. Tavares : Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8:1–15, 2003.

[LSCO+04] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl et H.-P. Seidel : Differential coordinates for interactive mesh editing. *Dans Proceedings of Shape Modeling International*, pages 181–190. 2004.

[LG00] X. Liu et R. Gadh : Virtualsketcher : Skeleton based 3D direct conceptual shape design in a virtual reality environment. URL ftp://ftp.computer.org/MAGS/CG&A/mms/111776.pdf. 2000.

[Loo87] C. Loop : Smooth subdivision surfaces based on triangles. Mémoire de D.E.A., University of Utah, 1987.

[LC87] W. E. Lorensen et H. E. Cline : Marching cubes : A high resolution 3D surface construction algorithm. *Dans Proceedings of ACM SIGGRAPH, papers*, 1987.

[Luc77] L. B. Lucy : A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024, 1977.

[Lé08] B. Lévy : *Géométrie Numérique*. Thèse de doctorat, INPL, France, 2008. HdR Habilitation thesis.

[LPRM02] B. Lévy, S. Petitjean, N. Ray et J. Maillot : Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, 2002.

[MPS08] J. Manson, G. Petrova et S. Schaefer : Streaming surface reconstruction using wavelets. *Computer Graphics Forum*, 27(5):1411–1420, 2008. Proceedings of SGP.

[MKB+10] S. Martin, P. Kaufmann, M. Botsch, E. Grinspun et M. Gross : Unified simulation of elastic rods, shells, and solids. *Dans Proceedings of ACM SIGGRAPH, papers*, 2010.

[MGP07] N. J. Mitra, L. Guibas et M. Pauly : Symmetrization. *Dans Proceedings of ACM Transactions on Graphics*, volume 26, pages #63, 1--8, 2007.

[MCG03] M. Müller, D. Charypar et M. Gross : Particle-based fluid simulation for interactive applications. *Dans Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 154–159. ACM, 2003.

[MG83] J. J. Monaghan et R. A. Gingold : Shock simulation by the particle method SPH. *Journal of Computational Physics*, 52:374 – 389, 1983.

[Mon92] J. J. Monaghan : Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30:543–574, 1992.

[ML85] J. J. Monaghan et J. C. Lattanzio : A refined particle method for astrophysical problems. *Astronomy and Astrophysics*, 149:135–143, 1985.

[MDM+02] M. Müller, J. Dorsey, L. McMillan, R. Jagnow et B. Cutler : Stable real-time deformations. *Dans Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–54, New York, NY, USA, 2002. ACM.

[NISA07] A. Nealen, T. Igarashi, O. Sorkine et M. Alexa : Fibermesh : designing freeform surfaces with 3D curves. *ACM Trans. Graph.*, 26(3) :41, 2007.

[NMK+06] A. Nealen, M. Müller, R. Keiser, E. Boxerman et M. Carlson : Physically based deformable models in computer graphics. *Comput. Graph. Forum*, 25(4):809–836, 2006.

[OBH02] J. F. O'Brien, A. W. Bargteil et J. K. Hodgins : Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3) :291–294, 2002.

[Oud10] H. Oudin : *Introduction à la plasticité*. École Centrale de Nantes, 2010.

[PZvBG00] H. Pfister, M. Zwicker, J. van Baar et M. Gross : Surfels : surface elements as rendering primitives. *Dans Proceedings of ACM SIGGRAPH*, pages 335–342, New York, NY, USA, 2000.

[PKC08] A. Pihuit, P. Kry et M.-P. Cani : Hands-on virtual clay. *Dans* G. X. Spagnuolo M., Cohen-Or D., éditeur : *Proceedings of IEEE International Conference on Shape Modeling and Applications*, pages 267–268, New-York, 2008. Shape Modeling Interface (SMI).

[PB00] D. Piponi et G. Borshukov : Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *Dans Proceedings of ACM SIGGRAPH*, pages 471–478, New York, NY, USA, 2000.

[PS96] K. Pulli et M. Segal : Fast rendering of subdivision surfaces. *Dans Proceedings of Eurographics workshop on Rendering techniques*, pages 61–70, London, UK, 1996.

[Ray03] N. Ray : *Génération d'atlas de texture multi-résolution*. Thèse d'université, INPL, 2003.

[RLL+06] N. Ray, W. C. Li, B. Lévy, A. Sheffer et P. Alliez : Periodic global parameterization. *ACM Trans. Graph.*, 25(4):1460--1485, 2006.

[Ree83] W. T. Reeves : Particle systems –- a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, 1983.

[RV09] M. Richtsfeld et M. Vincze : Point cloud segmentation based on radial reflection. *Dans Proceedings of Conference on Computer Analysis of Images and Patterns*, pages 955–962, Berlin, Heidelberg, 2009.

[Ris91] J.-J. Risler : *Méthodes mathématiques pour la CAO*. Masson, 1991.

[RB08] I. D. Rosenberg et K. Birdwell : Real-time particle isosurface extraction. *Dans Proceedings of the Symposium on Interactive 3D graphics and games*, pages 35–43, New York, NY, USA, 2008.

[SW04] S. Schaefer et J. Warren : Dual marching cubes : Primal contouring of dual grids. *Dans Proceedings of the Computer Graphics and Applications*, pages 70–76, Washington, DC, USA, 2004.

[SWSJ07] R. Schmidt, B. Wyvill, M. C. Sousa et J. A. Jorge : Shapeshop : sketch-based solid modeling with blobtrees. *Dans Proceedings of ACM SIGGRAPH*, pages 43, New York, NY, USA, 2007.

[SGW06] R. Schmidt, C. Grimm et B. Wyvill : Interactive decal compositing with discrete exponential maps. *Dans Proceedings of ACM SIGGRAPH*, pages 605–613, New York, NY, USA, 2006.

[SS10] R. Schmidt et K. Singh : meshmixer : an interface for rapid mesh composition. *Dans Proceedings of ACM SIGGRAPH*, pages 1–6, New York, NY, USA, 2010.

[SP86] T. W. Sederberg et S. R. Parry : Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

[Sei09a] M. Seiler : A particle-based volumetric sculpting tool. Mémoire de D.E.A., ETH Zurich, Department of Computer Science, 2009a.

[Sei09b] M. U. Seiler : A particle-based volumetric sculpting tool. Mémoire de D.E.A., Eidgenössische Technische Hochschule Zürich, Department of Computer Science, Institute of Visual Computing, 2009b.

[Set96] J. A. Sethian : A fast marching level set method for monotonically advancing fronts. *Dans Proceedings of the National Academy of Sciences*, pages 1591–1595, 1996.

[SLS+06] A. Sharf, T. Lewiner, A. Shamir, L. Kobbelt et D. Cohen-Or : Competing fronts for coarse-to-fine surface reconstruction. *Dans Proceedings of Eurographics*, 2006.

[SS07] C. Shen et A. Shah : Extracting and parametrizing temporally coherent surfaces from particles. *Dans Proceedings of ACM SIGGRAPH, sketches*, pages 66, New York, NY, USA, 2007.

[SYBF06] L. Shi, Y. Yu, N. Bell et W.-W. Feng : A fast multigrid algorithm for mesh deformation. *ACM Trans. Graph.*, 25(3):1108–1117, 2006.

[SZT+07] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao et B. Guo : Mesh puppetry : cascading optimization of mesh deformation with inverse kinematics. *Dans Proceedings of ACM SIGGRAPH*, pages 81, New York, NY, USA, 2007.

[Smi78] A. R. Smith : Paint. Rapport technique, New York Institute of T echnology, 1978.

[Smi79] A. R. Smith : Painting tutorial notes. *Dans Proceedings of ACM SIGGRAPH*, 1979.

[SP09] B. Solenthaler et R. Pajarola : Predictive-corrective incompressible sph. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 1–6, New York, NY, USA, 2009.

[SA07] O. Sorkine et M. Alexa : As-rigid-as-possible surface modeling. *Dans Proceedings of the Symposium on Geometry Processing*, pages 109–116, 2007.

[Sta99] J. Stam : Stable fluids. *Dans Proceedings of ACM SIGGRAPH, papers*, pages 121–128, New York, NY, USA, 1999.

[SK00] P. S. Stanimirovic et N. P. Karampetakis : Symbolic implementation of leverrier-faddeev algorithm and applications. *Dans Proceedings of the Conference on Control and Automation*, pages 6, 2000.

[SD06] A. Stern et M. Desbrun : Discrete geometric mechanics for variational time integrators. *Dans Proceedings of ACM SIGGRAPH, courses*, pages 75–80, New York, NY, USA, 2006.

[SSP07] R. W. Sumner, J. Schmid et M. Pauly : Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80, 2007.

[ST92] R. Szeliski et D. Tonnesen : Surface modeling with oriented particle systems. *SIGGRAPH Comput. Graph.*, 26(2):185–194, 1992.

[Tat06] N. Tatarchuk : Practical parallax occlusion mapping with approximate soft shadows for detailed surface rendering. *Dans Proceedings of ACM SIGGRAPH*, pages 81–112, New York, NY, USA, 2006.

[TF88] D. Terzopoulos et K. Fleischer : Modeling inelastic deformation : viscolelasticity, plasticity, fracture. *Dans Proceedings of ACM SIGGRAPH, papers*, 1988.

[THM+03] M. Teschner, B. Heidelberger, M. Müller, D. Pomeranets et M. Gross : Optimized spatial hashing for collision detection of deformable objects. *Dans Proceedings of Vision, Modeling, Visualization*, pages 47–54, 2003.

[Ton98] D. L. Tonnesen : *Dynamically coupled particle systems for geometric modeling, reconstruction and animation.* Thèse de doctorat, University of Toronto, 1998.

[TMC01] F. Triquet, P. Meseure et C. Christophe : Fast polygonization of implicit surfaces. *Dans Proceedings of WSCG*, 2001.

[vdLGS09] W. J. van der Laan, S. Green et M. Sainz : Screen space fluid rendering with curvature flow. *Dans* E. Haines, M. McGuire, D. G. Aliaga, M. M. Oliveira et S. N. Spencer, éditeurs : *Proceedings of the Symposium on Interactive 3D graphics*, pages 91–98. 2009.

[vGW94] A. van Gelder et J. Wilhelms : Topological considerations in isosurface generation. *ACM Trans. Graph.*, 13(4):337–375, 1994.

[Ver67] L. Verlet : Computer "experiments" on classical fluids. I. thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98, 1967.

[Ves04] M. Vesterlund : Simulation and rendering of a viscous fluid using smoothed particle hydrodynamics. Mémoire de D.E.A., University of Umea, 2004.

[WP09] R. Y. Wang et J. Popović : Real-time hand-tracking with a color glove. *Dans Proceedings of ACM SIGGRAPH*, pages 1–8, New York, NY, USA, 2009.

[WZS+06] R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng et B. Guo : Variational sphere set approximation for solid objects. *Vis. Comput.*, 22(9):612–621, 2006.

[WW01] J. Warren et H. Weimer : *Subdivision Methods for Geometric Design : A Constructive Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[Wil83] L. Williams : Pyramidal parametrics. *SIGGRAPH Comput. Graph.*, 17(3):1–11, 1983.

[Wil90] L. Williams : 3D paint. *Dans Proceedings of the Symposium on Interactive 3D Graphics*, pages 225–233, New York, NY, USA, 1990.

[WKB07] P. Wroblewski, M. Kopec et K. Boryczko : SPH - a comparison of neighbor search methods based on constant number of neighbors and constant cut-off radius. *Task Quaterly*, 3:273–283, 2007.

[YKH10] C. Yuksel, J. Keyser et D. H. House : Mesh colors. *ACM Transactions on Graphics*, 29(2):1–11, 2010.

[ZB05] Y. Zhu et R. Bridson : Animating sand as a fluid. *Dans ACM SIGGRAPH*, 2005.

[ZS00] D. Zorin et P. Schröder : Subdivision for modeling and animation. *Dans Proceedings of ACM SIGGRAPH*, 2000.

[ZPvBG01] M. Zwicker, H. Pfister, J. van Baar et M. Gross : Surface splatting. *Dans Proceedings of ACM SIGGRAPH*, pages 371–378, New York, NY, USA, 2001.

Résumé

La 3D s'impose comme un nouveau média dont l'adoption généralisée passe par la conception d'outils, accessibles au grand public, de création et de manipulation de formes tridimensionnelles quelconques. Les outils actuels reposent fortement sur la modélisation sous-jacente des formes, généralement surfacique, et sont alors peu intuitifs ou limitatifs dans l'expressivité offerte à l'utilisateur.

Nous souhaitons, dans ces travaux, définir une approche ne présentant pas ces défauts et permettant à l'utilisateur de se concentrer sur le processus créatif. En nous inspirant de l'utilisation séculaire de l'argile, nous proposons une approche modélisant la matière sous forme lagrangienne. Une forme est ainsi décrite par un système de particules, où chaque particule représente un petit volume du volume global.

Dans ce cadre lagrangien, la méthode *Smoothed Particle Hydrodynamics* (*SPH*) permet l'approximation de grandeurs physiques en tout point de l'espace. Nous proposons alors une modélisation de matériaux à deux couches, l'une décrivant la topologie et l'autre décrivant la géométrie du système global.

La méthode *SPH* permet, entre autres, d'évaluer la densité de matière. Ceci nous permet de définir une surface implicite basée sur les propriétés physiques du système de particules pour redonner un aspect continu à la matière.

Ces matériaux peuvent alors être manipulés au moyen d'interactions locales reproduisant le maniement de la pâte à modeler, et de déformations globales. L'intérêt de notre approche est démontrée par plusieurs prototypes fonctionnant sur des stations de travail standard ou dans des environnements immersifs.

Mots-clefs : modélisation volumique, 3D, *Smoothed Particle Hydrodynamics*, modélisation lagrangienne, simulation de matériau, algorithmique, reconstruction de surface, surface implicite, déformation locale, déformation globale, ajout de détails, sculpture, forme libre.

Abstract

3D is emerging as a new media. Its widespread adoption requires the implementation of user-friendly tools to create and manipulate three-dimensional shapes. Current softwares heavily rely on underlying shape modeling, usually a surfacic one, and are then often counter-intuitive or limiting.

Our objective is the design of an approach alleviating those limitations and allowing the user to only focus on the process of creating forms. Drawing inspiration from the ancient use of clay, we propose to model a material in a lagrangian description. A shape is described by a particles system, where each particle represents a small fraction of the total volume of the shape.

In this framework, the Smoothed Particle Hydrodynamics method enables to approximate physical values anywhere in space. Relying on this method, we propose a modeling of material with two levels, one level representing the topology and the other one describing local geometry of the shape.

The *SPH* method especially enables to evaluate a density of matter. We use this property to define an implicit surface based on the physical properties of the particles system to reproduce the continuous aspect of matter.

Those virtual materials can then be manipulated locally through interactions reproducing the handling of dough in the real world or through global shape deformation. Our approach is demonstrated by several prototypes running either on typical desktop workstation or in immersive environment system.

Keywords: volumic modeling, 3D, Smoothed Particle Hydrodynamics, lagrangian modeling, material simulation, algorithmic, surface reconstruction, implicit surface, local deformation, global deformation, surface details, sculpturing, free form.